LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

IO VI. 2ª ÉPOCA NÚMERO 67 UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L. 975 Ptas. • 995 Escudos portugueses • 5,86 € (IVA incluido)

Microsoft Windows 2000 Server

BASES DE DATOS
Aplicaciones de Bases de Datos en Delphi 5

VISUAL BASIC
Programación orientada a objetos

ENTORNOS DE DESARROLLO
Visual C++ y MFC

PROGRAMACIÓN WEB
Aplicaciones Web con acceso a BBDD basadas en Java

LINUX
MP3: Reproducción en nuestros programas

TELECOMUNICACIONES
Curso de WAP

DELPHI
Programación OpenGL en Delphi

HERRAMIENTAS
VisualCafé 4 Vs. JBuilder 3

CONTENIDO CD
- BeOS 5 Personal Edition
- Delphi 5 Enterprise
- WinRoute Pro 4.1a
- JDesignerPro 4.0
- BW Zip Compress 3.0.9.2
- MDaemon 3.02
- Voice Tools 6.0
- Nero 4.0.9.1
- Anonymity 4 Proxy v2.01
- InstallConstruct 3.3

00067
8 413042 303299

Sólo PROGRAMADORES LINUX

UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L.

AÑO III. Nº 18. TERCERA ÉPOCA P.V.P. 950 PTS. • 990 ESCUDOS PORTUGUESES • 5,7 € (IVA incluido)

CONTENIDO CD 38 APLICACIONES Y LIBRERÍAS

XFREE86 4.0: no por mucho madrugar...

SISTEMA OPERATIVO
Debian Potato 2.2

HERRAMIENTAS
Diseño con Gimp

UTILIDADES
Edición en LaTex

PROGRAMACIÓN
La Shell de Unix y el AWK

MULTIMEDIA
MP3 en Linux

SOFTWARE
Ofimática Linux

PROGRAMACIÓN GRÁFICA
Programación multimedia con SDL

REDES
Capacidades avanzadas de Samba

TODOS LOS MESES EN TU QUIOSCO



SÓLO PROGRAMADORES

Número 67
SÓLO PROGRAMADORES
es una publicación de
REVISTAS PROFESIONALES S.L.

Editor
Agustín G. Buelta
Coordinador Técnico
Eduardo De Riquer Frutos
Coordinadora de Redacción
Gema Romero Moreno-Manzanaro
Edición
Francisco Pino García
Gema Prados Mateos
Colaboradores
Constantino Sánchez, Juan Luis Ceada,
Javier Sanz, Adolfo Aladro,
Enrique de la Lastra, Marc Vaquer
Jordi Agost, Vicente A. Sánchez Werner
Maquetación y Tratamiento de Imagen
Paco Risco
Consultas técnicas
atecnica@virtualsw.es

Asesoría de Publicidad
Carmina Ferrer
Tel.: (91) 304 78 46
Mariano Sánchez (Barcelona)
Tel.: (93) 322 12 38

Suscripciones
Rosa Tabares
Tel. (91) 304 87 64 Fax: (91) 327 13 03

Impresión
I. de Impresión
Distribución
Motorpress Ibérica



Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 31-7-2000
Precio en Canarias, Ceuta y Melilla:
938 ptas. sin I.V.A.
Con sobretasa aérea: 975 ptas.



Asociación Española de Editoriales
de Publicaciones Periódicas

EDITORIAL

¿Cuándo te cambias a Windows 2000?

Esta pregunta tiene, además del evidente propósito de llamar la atención de nuestros lectores, el objetivo de poner sobre la mesa algunos puntos que toda la tormenta desatada sobre *Micrososft* desde su condena y la subsiguiente caída en la bolsa y de la bolsa, apenas se han tratado y que nosotros pretendemos abordar en estas páginas durante varios meses.

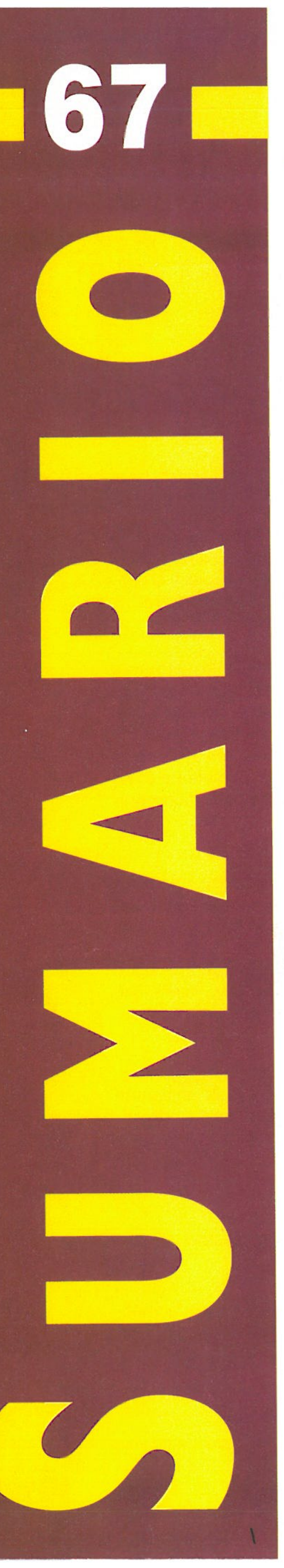
Procuraremos centrarnos en los aspectos técnicos, porque los otros ya están siendo tratados hasta la saciedad. Y dejaremos claro que no pretendemos defender a *Microsoft,* sino que se trata de enfrentar el hecho de que *Windows* es el sistema operativo con el que trabajan la inmensa mayoría de los *PC's* del mundo, y que *Windows 2000* aporta importantes ventajas con el nuevo *Directorio Activo*, entre ellas la importante mejora que supone el no tener que reiniciar después de cada actualización de *software* o la incorporación de *hardware*.

La verdad es que no es concebible un sistema operativo serio y que quiere ser un firme contendiente en el mundo de la gran empresa, sin unas mínimas prestaciones como la citada. Y por fin hemos llegado a donde íbamos: se trata de saber si *Windows 2000* es un sistema operativo consistente o no. La respuesta no puede darse *a priori*. Nosotros iremos publicando artículos de análisis como el que incluimos este mes sobre la vesión *Server*. Esperamos que nuestros lectores nos hagan llegar sus experiencias y opiniones al respecto.

Entre los problemas que algunos profesionales han detectado está la pesadez/lentitud de *Windows 2000 Professional*. Otro podría ser el deficiente soporte multimedia. Y no hay que olvidar el problema que se está planteando con las múltiples versiones de *W2000* que existen o existirán en función de la máquina sobre la que trabajen, los problemas de compatibilidad que ello supone no parece que sean pocos.

Queridos lectores, se abre la sesión. Esperamos ansiosos vuestras aportaciones, sobre todo las basadas en la experiencia.

Para terminar, respondemos a la pregunta del titular: como usuario doméstico no encuentro razones para cambiarme, más bien las tengo para no cambiarme. ¿Estamos todos de acuerdo?



SÓLO PROGRAMADORES

30 Delphi

PROGRAMACIÓN OPENGL EN DELPHI (II)



En este número, continuamos con la serie abierta el mes pasado, dedicada a la programación gráfica con el estándar *OpenGl* para la inserción de imágenes *3D*. En esta ocasión, aprenderemos a utilizar las funciones *OpenGl* destinadas a operar sobre los objetos mediante el cálculo matricial, con el fin de que el usuario pueda manipular a su antojo el objeto creado. ¡Pero no os asustéis, para hacer esto, no hace falta ser un eminente matemático! Bastará con poseer algunos conocimientos elementales referidos al concepto de matriz,... del trabajo duro, los cálculos algorítmicos, se encarga el propio programa.

44 Linux

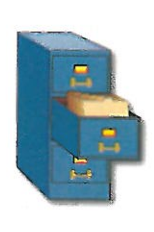
MP3 EN LINUX (y III): AÑADIENDO REPRODUCCIÓN MP3 A NUESTROS FICHEROS (II)



ion muchas las cosas que *MP3* nos permite acer: desde un simple reproductor de línea le comando, o añadir una banda sonora a uestro juego favorito, hasta realizar un proesamiento musical o sincronizar eventos con núsica. Vamos a hablar en esta ocasión de cóno valernos de *SDL* y *SMPEG* para conseguir ina perfecta reproducción *MP3* en nuestros rogramas. Incluiremos nuevas prestaciones ue nos permitan aprovecharnos de la arquitectura multihilo.

52 Bases de datos

BASES DE DATOS EN DELPHI 5 (IV)



ras habernos ocupado en números anteriores e los componentes de acceso y la edición de atos, y haber desarrollado una aplicación práctica, vamos a estudiar en este número la labor on diferentes sistemas gestores de bases de atos, además de adentrarnos en los nuevos omponentes *VCL* y *SQL*, como paso previo ntes de dedicarnos plenamente a la parte ráctica, con la conexión a los sistemas gestoes más comunes, como *Oracle* o *Interbase*.

58 Visual Basic

PROGRAMACIÓN ORIENTADA A OBJETOS (II)



espués de haber presentado las novedades de *isual Basic 7* tenemos que continuar esta serie obre programación orientada a objetos con lo ue se puede hacer actualmente, es decir, tenenos que hablar de *Visual Basic 6*. Empezaremos or las aclaraciones conceptuales de la programación de objetos y sus fases, desde el objeto nismo y la construcción de clases, hasta el tratamiento de herencia y polimorfismo.

66 Entornos de desarrollo

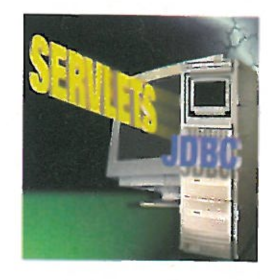
VISUAL C++ Y MFC (IV)



En el último número aprendimos a controlar la información dentro de las cajas de texto incluidas en las ventanas o formularios por medio de código, tablas, y funciones; en esta ocasión vamos a detenernos en la creación y utilización de los menús de opciones, organizados en órdenes, submenús y separadores, en nuestros programas, para dotarlos de una funcionalidad óptima.

72 Programación Web

APLICACIONES WEB CON ACCESO A BBDD BASADAS EN JAVA (V)



Tras ocuparnos en el último número de optimizar nuestra aplicación *Web* por medio de las distintas configuraciones a nuestro alcance, vamos a aprender ahora a exprimirla. El objetivo no es ni más ni menos que sacar el mayor rendimiento de nuestra aplicación en el momento de acceder a las fuentes de datos, valiéndonos de una nueva interfaz que nos permite trabajar con sentencias *SQL* preparadas y con parámetros.

78 Libros

En esta sección, os informamos de las últimas novedades editoriales que pensamos os pueden ser de mayor interés. Con cada referencia, una breve reseña acompañada de información de precio, idioma y editorial, así como una útil valoración de nivel de dificultad.

80 Ayuda al lector

No dudéis en mandarnos vuestras dudas y consultas relacionadas con los contenidos de la revista, o cualquier otro problema que os "lleve de cabeza". Sólo tenéis que enviarnos un *E-mail* a *solop@virtualsw.es*.

JAVA EXPO YA ESTÁ EN MARCHA

Java Expo 2000, lugar de encuentro de los profesionales y curiosos del mundo *".com"* se celebrará durante los próximos 16 y 17 de mayo en el *Palacio Municipal de Congresos de Madrid*. La tercera edición de esta feria reunirá a más de 100 representantes de empresas, emprendedores, e inversores relacionados con uno de los sectores de la comunicación y la informática de mayor dinamismo.

El evento, impulsado por *Sun Microsystems*, estará dividido en dos jornadas, dedicada la primera a las nuevas tecnologías (*Java* corporativo, lenguaje *XML*, dispositivos de consumo y seguridad), y la segunda a las empresas y el *negocio ".com"*. Se prevé la participación de más de 100 empresas y la celebración de 60 conferencias en diferentes sesiones paralelas, con horario de 9 de la mañana a 6 de la tarde.

La exposición, de libre acceso, ocupará más de 2.500 m², donde tendrán especial protagonismo las empresas españolas, y se prestará atención a los medios de comunicación en la *Red* (*Media.com*). *Java Expo* cuenta este año con un importante presupuesto de 100 millones de pesetas, y está patrocinada por las empresas *Cisco, Telefónica Data, SAP, I-Plant* y *Oracle*.



Javaexpo tiene su réplica virtual en la Red.

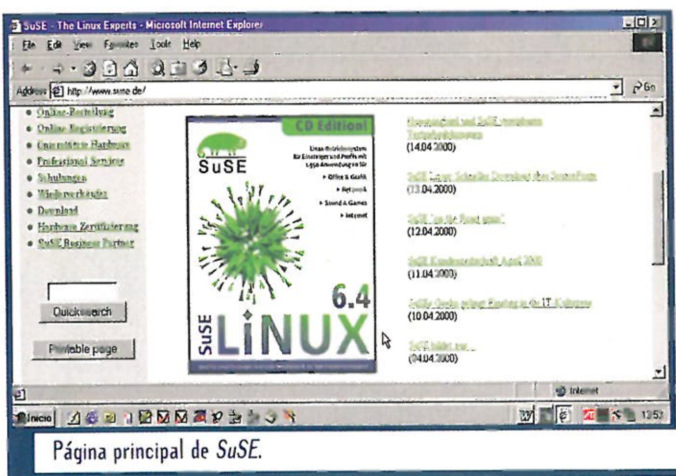
Como novedad, este año *Java Expo 2000* tendrá su réplica virtual, y podrá seguirse en la *Red*, en la dirección *www.sun.es/javaexpo*. Los interesados en obtener más información también pueden diregirse a esta página *Web* y hacerse con el programa de conferencias.

SUSE PARA LINUX INTEL, CON TECNOLOGÍA ENLIGHTEN

SuSE Linux AG y *Enlighten Software Solutions* han anunciado una alianza estratégica, por la que *Enlighten* dotará al proveedor de *Linux* de su tecnología de monitorización y generación de archivos en la última versión de *SuSE* para *Intel*.

De esta forma, los clientes de *SuSE* podrán instalar el sistema operativo con el agente de monitorización *Enlighten*, generando archivos en condiciones operativas y de sistema críticas que afecten al procesador, memoria, *hardware, software*, y red.

Asimismo, el agente *Enlighten* indica a los usuarios de *SuSE* la página *Web* donde encontrarán información sobre las más de 80 pruebas que efectúa el agente, y donde podrán registrarse para obtener la licencia *Linux*, o actualizar el producto, además de poder utilizar la versión completa de *EnlightenDSM*, solución que permite monitorizar y gestionar desde un solo punto sistemas *Linux, Unix* y *Windows*. Más información en: *www.enlightenDSM.com*.



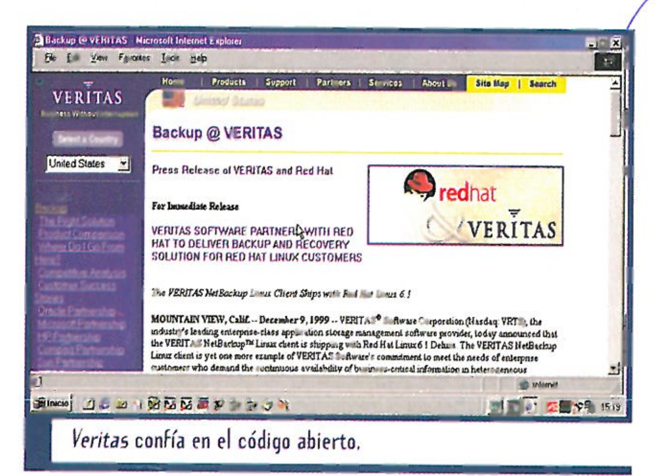
Página principal de *SuSE*.

INCOMPATIBILIDAD DITTO Y WINDOWS 2000

La compañía *Tecmar*, fabricante de periféricos de cinta magnética, utilizados para el almacenamiento masivo y generación de copias de seguridad, advierte de su incompatibilidad con *Windows 2000*, pues no incluye los controladores necesarios.

Aún no se tienen datos referentes a otros fabricantes de este tipo de dispositivos internos o de puerto paralelo, pero *Tecmar* ya ha anunciado que ofrecerá soporte para *Windows 2000* en las próximas versiones de sus productos.

VERITAS SE ABRE A LINUX



Veritas confía en el código abierto.

La compañía de *software* para gestión y almacenamiento *Veritas*, ha anunciado la oferta para este año de sus productos *Veritas Volume Manager, Fyle Systems, Cluster Server*, y su versión servidor *NetBackup* para el sistema operativo *Linux*, al tiempo que se compromete con una política de código abierto.

En este mismo contexto, *Veritas* ha alcanzado un acuerdo con *Red Hat*, para acelerar la disponibilidad de sus productos en ese sistema operativo.

Para más información se puede acudir a la siguiente dirección: *www.veritas.com.*

PANDA PLATINUM PARA WINDOWS 2000

Ya está lista la nueva versión del antivirus *Panda* dirigida a *Windows 2000*. Entre sus novedades, destacan el nuevo *Sentinel*, capaz de optimizar la protección residente, con unos resultados de baja carga en el rendimiento de la *CPU*, y el análisis heurístico avanzado para *win32* y *macro*.

Además, *Panda Platinum Windows 2000* aporta la certificación *ICSA.net*, que garantiza la detección y desinfección del 100% de los virus en la *Wild*.

PHOTO-PAINT 9 EN LINUX

Corel Corporation ha anunciado que ofrecerá una versión *freeware* de su programa de diseño gráfico *PHOTO-PAINT 9* junto con *Coreldraw 9 Graphics Suite* para *Linux*, a partir del próximo verano. Esta versión tendrá un precio similar a la de *Windows*.

Dicha aplicación ofrece una interfaz sencilla y completa, compatible con los diferentes formatos y una solución guiada de publicación *Web* y *PDF*.

ECONOMIC DATA PRESENTA SU NUEVA HERRAMIENTA DE CONTROL REMOTO

Economic Data ha lanzado *Proxy Remote Control Gateway*, una herramienta que integra control remoto, monitorización de pantallas, y un sistema de grabación y reposición de pantallas, para facilitar el control de calidad y documentación de actividades en un *call-center*.

PRESENTACIÓN DE TARANTELLA ASP EDITION

Tarantela ASP Edition, el nuevo programa de *SCO* para proveedores de servicios, combina la funcionalidad de un *software* de capacitación *Web*, con un conjunto de formación, soporte y actualizaciones de mantenimiento automáticas, destinado a facilitar a los proveedores de servicios de aplicación la optimización de la gestión y despliegue centralizado de aplicaciones *Web*.

Este producto incorpora, además, la monitorización de uso y modalidades de facturación, que permiten a los *ASP's* analizar e informar del uso de la aplicación a sus clientes, con un tipo de licencia de pago por tiempo de uso.

Junto a *Tarantella ASP Edition, SCO* presenta también el programa *Tarantella ASP Connect*, dirigido a interconectar a los *OEM's, ISV's* y *ASP's* dentro de una red técnica y educacional en la que puedan participar en seminarios, foros y eventos destinados al uso y desarrollo del modelo de negocio *ASP*.

IBM PRESENTA LA BETA DE TOPPAGE PARA LINUX

IBM avanza en su política de apoyo a *Linux*, tras el anuncio de una versión de *TopPage*, su *software* de edición de páginas *Web*, disponible para *Linux*. Entre las prestaciones de este *software*, encontramos la edición en modo *WISIWYG (What you see is what you get)*, modo *HTLM*, visión previa, edición múltiple, soporte *HTLM 4.0, CSS 1*, soporte *DHTLM*, gestión de sitios *Web* integrada, soporte para la creación de canales compatibles con *Netscape* e *Internet* y soporte de marcos.

Esta versión, plenamente funcional, se puede descargar gratuitamente de la siguiente dirección: *http://www.ibm.com/jp/topage.*



El nuevo procesador *Intel Pentium* III a 1 *GHz* llega al mercado.

EL ACCESO INALÁMBRICO A LA RED SUPERARÁ A LA CONEXIÓN POR CABLE

La empresa *IDC*, especialista en los estudios de evolución y desarrollo de las nuevas tecnologías, ha elaborado un informe que prevé un extraordinario aumento de los internautas que acceden a la *Red* a través del protocolo *WAP* en los próximos años. Según *IDC*, a mediados del 2001, todos los *PC* portátiles, así como los teléfonos móviles, soportarán este protocolo.

Las previsiones de esta empresa se apoyan en el hecho de que el número de usuarios de telefonía móvil actualmente supera ampliamente al de usuarios que se conectan a la *Red on-line*, y los operadores podrán abaratar los costes de los servicios ofrecidos a sus clientes, que se convierten de esta forma en internautas potenciales.

ENTORNOS DE PROGRAMACIÓN

ENTORNOS DE PROGRAMACIÓN

ASM EDIT 2.8

Editor de ficheros fuente en lenguaje ensamblador que permite crear y mantener el código, y la documentación correspondiente. Puede editar ficheros de hasta 64 *Kb*. Es capaz de emular la mayoría de los comandos de *WordStar*, usar los caracteres de fin de fichero, deshabilitar en pantalla la línea de estado, etc.

BEOS 5 PERSONAL EDITION

Potente sistema operativo capaz de sustituir a *Microsoft Windows*. Es fácil de instalar y no interfiere con el sistema operativo que tenga instalado el equipo. Incluye un editor de texto, un navegador, un cliente de correo electrónico, un cliente *FTP*, una utilidad de compresión en formato *ZIP*, un reproductor multimedia, juegos y otras utilidades. Soporta los formatos de ficheros más habituales, que han sido creados en otros sistemas operativos. Aunque se trata de una versión limitada, permite apreciar la potencia de la versión completa.

DELPHI 5 ENTERPRISE

Herramienta de desarrollo rápido de aplicaciones (*RAD, Rapid Application Development*) para *Internet*, aplicaciones distribuidas y de bases de datos. Simplifica la integración de *Windows* con servidores *Web*, navegadores y bases de datos. Incluye un asistente para crear sistemas extensibles que soporten comercio electrónico usando *HTML* 4 y *XML*, y opciones para agilizar la distribución de aplicaciones *HTML*. Es necesario registrarse para obtener un código de acceso que permita realizar la instalación de la versión de evaluación.

M68000 INTEGRATED DEVELOPMENT ENVIRONMENT 1.0

Entorno de desarrollo para equipos con procesador *M68000*. Edita, compila y ensambla programas escritos para micro-ordenadores *M68000*. Soporta programas en *Assembler*, *C*, o *Pascal* que se ejecutan sobre un simulador interno o sobre un sistema externo conectado al ordenador a través de un puerto serie.

SUPERIDE 1.4.2.168

Entorno de desarrollo integrado que trabaja con cualquier compilador y lenguaje. Incluye plantillas de código, una interfaz *MDI*, sintaxis *highlighting*, definición de herramientas, corrección de código, capturas en modo consola, y capacidad para guardar el código destacado en formato *HTML* o *RTF*.

VEDIT PLUS 5.16.3

Potente editor de texto que puede editar rápidamente cualquier fichero *Windows*, *DOS*, *UNIX*, o *MAC* de texto. Es capaz de trabajar con grandes ficheros binarios, caracteres gráficos y de control, y registros de bases de datos de longitud fija o variable.

LENGUAJES

VISUAL BASIC

BW ZIP COMPRESS 3.0.9.2

Control *ActiveX* (*OCX*) para trabajar con ficheros *ZIP*. Puede crear, actualizar, modificar, probar, listar, reparar y extraer ficheros *ZIP*. Soporta ficheros *ZIP* encriptados, nombres largos (con soporte *UNC*) y puede cambiar la prioridad de la *CPU* de compresión o descompresión.

CHECKLIST BOX ACTIVEX/VBX 2.5 BUILD 8

Control para generar casillas de comprobación que permite crear listas diferentes en las que los usuarios puedan comprobar *items* individuales. Incluye opciones para personalizar las fuentes, los colores, etc.

MODEM CID 1.1

Control *ActiveX* que hace un seguimiento de las llamadas de teléfono entrantes. Incluso antes de contestarlas, muestra información sobre su emisor como el nombre, el número de teléfono, si se encuentra fuera de área o si ha activado la opción de ocultar el número.

JAVA

JAVA STUFF 2.0

Colección de *scripts* de *JavaScript* para mejorar páginas *Web*. Entre otras categorías, incluye *scripts* para alertas, *cookies*, *frames*, imágenes, *IP*, *links*, el ratón, el sistema operativo, eliminar *Pop-Up*, el tamaño de la pantalla, sonido, barra de estado, etc.

JDESIGNERPRO 4.0

Crea bases de datos interactivas en lenguaje *Java*. Permite incluir las bases de datos en *Internet* y genera automáticamente el código *Java* y *SQL*. El sistema ofrece un flexible control de acceso, capacidad para crear sistemas configurables de menú, creación automática de código *Java*, un gestor de pantallas

Java, un explorador de métodos, gestión de proyectos con seguimiento de varios programadores, un navegador interno, una interfaz de correo electrónico y un motor de bases de datos *Java*.

JET DEPLOYMENT ENVIRONMENT 0.5 BETA

Compilador de código *Java* en lenguaje nativo. El compilador *JET* lee ficheros de clases *Java* creados en cualquier entorno de desarrollo Java, los compila utilizando potentes técnicas de optimización especialmente adaptadas a *Java*, los enlaza con librerías de alto rendimiento y produce un fichero ejecutable convencional.

HTML

EASY XML 1.0

Sencillo editor *XML* que agiliza la creación de aplicaciones *XML*. Es una herramienta adecuada para desarrolladores de sitios *Web*, consultores, analistas de bases de datos y todos aquellos que necesiten conocer el mundo del comercio electrónico.

FAQ-MAN 1.03

Ayuda a crear sistemas de ayuda en línea para sitios *Web*. Los sistemas creados tienen la forma de los conocidos *FAQ* (*frequently asked questions*). Únicamente hay que introducir las preguntas, las respuestas, organizarlas en categorías, añadir la información del sitio y cargar los ficheros.

SCRIPTWORX 4.0

Completo editor *HTML* que soporta los últimos estándares de *Internet*. Incluye un servidor *Web*, un cliente *FTP*, un generador de documentos *ColdFusion*, asistentes, herramientas de creación de *CSS* y alrededor de 100 plantillas.

OTROS

DELPHI VCL EXTENSIONS 2.75

Colección de componentes, objetos y rutinas para *Delphi* y *C++ Builder*. Los componentes pueden ser utilizados para la edición de datos, manipulación de rejillas, filtros de registro, introducción de datos, etc.

DIGITAL MUSIC CONTROL 2 V1.0

Control *ActiveX* capaz de reproducir varios formatos de sonido digital. Reproduce ficheros *IT*, *MO3*, *MOD*, *MP3*, *MTM*, *S3M*, *WAV* y *XM*. Soporta tarjetas aceleradoras *DirectSound* si se encuentran disponibles. Ofrece la posibilidad de establecer la frecuencia, el volumen y otros parámetros durante la reproducción.

MULTIMEDIA CONVERSION LIBRARY 2.0.4

Añade a nuestros desarrollos, funciones de conversión automática de imágenes a vídeo. Puede importar cualquier tipo de fichero de imágenes incluyendo *GIF, JPG, JPEG, PNG, TGA* y *TIF*. Exporta a los formatos de vídeo *AVI, FLC, FLI, HAV, M2V* y *MPG*.

VIRTUAL PRINT ENGINE PROFESSIONAL EDITION 3.1

Control *Runtime* para generar informes, presentaciones, gráficos y diagramas desde cualquier aplicación. Es un control *WYSIWYG* que incluye 21 tipos de códigos de barras. Permite al usuario añadir texto e imágenes (en formato *TIFF, JPEG, GIF, PNG, PCX, BMP, WMF, EMF* o *DXF*) y crear líneas, polígonos, casillas, elipses, tramas, etc.

VOICE TOOLS 6.0

Añade a las aplicaciones funciones de reconocimiento de voz. Se trata de una colección de 10 controles *ActiveX* y recursos de librerías especiales que permiten desde la activación por voz de objetos estándar de *Visual Basic*, realizar dictados, navegar utilizando la voz, hasta convertir texto a voz.

HERRAMIENTAS Y UTILIDADES

BUGGIT 2.09.2

Hace un seguimiento y gestiona errores durante el desarrollo de un programa. Incluye opciones para editarlos, imprimir informes, generar gráficos y administrar bases de datos de proyectos. Ofrece un número ilimitado de bases de datos multiusuario, capaces de trabajar con un equipo de desarrollo completo.

ENIGMA 1.30

Entorno de desarrollo integrado para crear y editar recursos. La diferencia entre *Enigma* y otros entornos de desarrollo es que es posible utilizar *Enigma* para crear y editar recursos como diálogos, menús, barras de herramientas, barras de estado, etc.

GOLDWAVE 4.16

Editor de ficheros de sonido digital. Extrae sonidos de ficheros de *CD´s* de audio, vídeo, aplicaciones *Java* y sitios *Web*, y puede guardarlos en disco. Soporta una gran variedad de formatos e incluye numerosos efectos para mejorar los sonidos o para crear otros nuevos.

INSTALLCONSTRUCT 3.3

Herramienta de desarrollo para crear asistentes de configuración de programas, desinstaladores y paquetes que pueden ser distribuidos. Sus opciones permiten configurar el proceso de instalación como se desee, de forma que se pueda iniciar automáticamente, solicitando una *password*, una firma, etc.

NERO 4.0.9.1

Utilidad que puede grabar datos y audio en discos *CD-R* y *CD-RW*. Puede trabajar con todos los formatos estándar de *CD-ROM* y convertir, de forma dinámica, los nombres de ficheros a formato *ISO*. Ofrece

soporte para *HFS*, *ISO/HFS-Hybrid*, *OFAS*, *UDF* y *UDF-Bridge*.

TEST CASE MANAGER 2.03

Herramienta de gestión de pruebas de *software*. Permite organizar las pruebas en protocolos y registrar los datos de su ejecución. Las pruebas son escritas en un formato estándar y guardadas en el sistema. Una vez que son ejecutadas, permiten generar informes con los fallos encontrados.

VISUAL PROTECT 1.0

Herramienta de gestión de licencias, comercio electrónico y protección de *software*. *Visual Protect* protege cualquier fichero ejecutable de 32 *bits* para evitar su copia y distribución ilegal.

REDES

ALCHEMY EYE 1.5

Utilidad de administración de redes que permite monitorizar servidores en entornos *LAN* y *WAN*. Cuando el estado del servidor cambia, *Alchemy Eye* puede grabar la descripción del evento en un fichero *LOG*, que se guarda en formato *HTML*.

ANONYMITY 4 PROXY V2.01

Servidor *proxy* que permite navegar por *Internet*, asegurando la privacidad y manteniendo el anonimato. Proporciona herramientas para localizar, probar y entrar en servidores *proxy* abiertos.

DEVICELOCK 4.1.03

Controla el acceso a discos duros y otros dispositivos de la red. Permite a los administradores de redes asignar permisos para acceder a puertos *COM*, *LPT*, *CD-ROM's*, etc.

ESSENTIAL NETTOOLS 2.1

Programa de administración de redes. Incluye utilidades de seguridad y diagnóstico, un escáner de *NetBIOS*, un monitor de conexiones externas a los recursos compartidos de red, una herramienta para conectar rápidamente con recursos de *Windows* 95/98, etc.

MDAEMON 3.0.2

Excelente servidor *SMTP* y *POP3*. Permite integrar fácilmente en una red, funciones de correo electrónico a través de *Internet*. Incluye un *firewall* de correo electrónico, un editor de cuentas, alias de dominios y cuentas, listas de correo, soporte *RAS*, un programador de tareas, opciones para evitar la recepción de mensajes publicitarios y funciones de control remoto.

DOCUMENTACIÓN/ TUTORIALES

ELECTRONIC COMPUTER TUTOR 2000

Tutorial de información general sobre ordenadores. Es una aplicación que ofrece información sobre diferentes temas relacionados con el uso y mantenimiento de los equipos.

HTML TUTOR: BEGINNERS COURSE 2.0

Tutorial de introducción a la programación en lenguaje *HTML* en formato *WORD*. Este tutorial está dirigido a aquellas personas que están comenzando a crear sitios *Web*.

INTRODUCTION TO TCP/IP PROGRAMMING

Tutorial sobre desarrollo de programas para *Internet*. Los temas que aborda han sido creados de modo que cualquiera, con una pequeña experiencia en programación, pueda entenderlos.

LEARN ELECTRONICS 2.04

Tutorial interactivo sobre conceptos y aspectos generales de electrónica. Primero muestra la información y luego hay que realizar un pequeño examen para comprobar que se han entendido los conceptos.

IMPRESCINDIBLES

ANTIVIRUS

AntiViral Toolkit Pro 3.0.129
AVP Virus Enciclopedia
eSafe Desktop 2.2.24
McAfee VirusScan 4.0.3
Panda Antivirus 6.16.00 Platinum
Quick Heal 5.23

GRÁFICOS

Icon Bank 4.0
IconForge 4.6
MicroAngelo 98 v4.77
Paint Shop Pro with Animation Shop 6.02
Reptile 2.01
SureThing CD Labeler 2.0
ThumbsPlus 4.10
Ultra Fractal 2.04
Xara WebStyle 1.2

INTERNET

AutoWinNet 6.0 Beta 1
Copernic 2000 v4.1
Cuentapasos 3.77
CuteFTP 4.0
Dial-Up Magic 1.8
Eudora Pro 4.3.1
Free Agent 1.21
GetRight 4.2
Go!Zilla 3.5
Guardián 1.1
HotDog Professional 5.5
ICQ 99b beta 3.19 build 2569 Rev.A
i.Share 3.5
Jet Color 1.20
Mass Downloader 1.2.87
MIRC 32 5.7
SubmitWolf ES 4.02.002 (Añadir Pro)
TS-ImageMapper 2.00
URL Organizer 2.3.3
WebZIP 3.60
WinGate 3.0.5
XMLwriter 1.1

MULTIMEDIA

AudioCatalyst 2.01
CDH Media Wizard 4.55
COWON Jet-Audio 4.7
MusicMatch Jukebox 5.0
Sonique 1.51
WinAMP 2.61

NAVEGADORES

Internet Explorer 5.01
NeoPlanet 5.1
Netscape Communicator 4.72
Opera 3.62

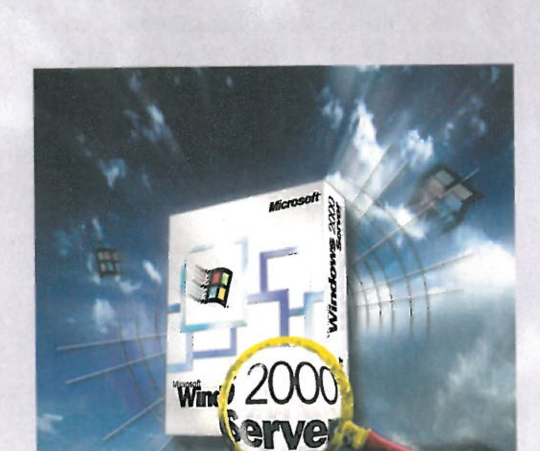
PROGRAMACIÓN

Decafe Pro 3.6
Hackman 4.02
Help & Manual 2.51
Java Development Kit 1.2.2-001
UltraEdit Professional Text/HEX Editor 7.10
Windows Registry Guide 1.3
WinHex 9.31

UTILIDADES

Adobe AcrobatReader 4.0
Babylon Translator Spanish 2.2
CallCenter 3.5.8
DirectX 7.0a
Emergency Recovery System 9.05
SiSoft Sandra 2000.3.6.3
WebTrends Professional Suite 4.0c
Where Is It? 2.16
Windows Commander 4.03
WinZip 7.0

ATENCIÓN:
En caso de problemas con el CD-ROM envíelo por correo ordinario, a la atención del SERVICIO TÉCNICO DE SÓLO P., incluyendo en el interior del sobre sus datos personales, a la siguiente dirección:
C/ San Sotero, Nº 5, 1ª Planta, 28037 (Madrid)



Windows 2000 Server: La nueva generación

Marc Vaquer Crusat.
Técnico de Sistemas.

Ya ha llegado el futuro de los sistema operativos en cuanto a *Microsoft* se refiere. Aquí os ofrecemos los pros y los contras del tan aclamado y esperado *Windows 2000* en sus diferentes versiones.

INTRODUCCIÓN

Windows 2000 Server es un sistema operativo que se puede utilizar como servidor de red o *Web*, servidor de ficheros, impresoras, o aplicaciones.

Windows 2000 por fin soporta FAT32 y PnP

De la versión *Server*, una de las cuatro que estarán en el mercado, se rumorea que costó 41 millones de líneas de código. Nos detenemos en este artículo a explicar las novedades de este sistema operativo, sus características, así como la manera de efectuar su instalación. En definitiva, ofrecemos una visión detallada del producto, con el fin de que el usuario pueda valorar si vale la pena o no actualizarse.

LAS CUATRO VERSIONES

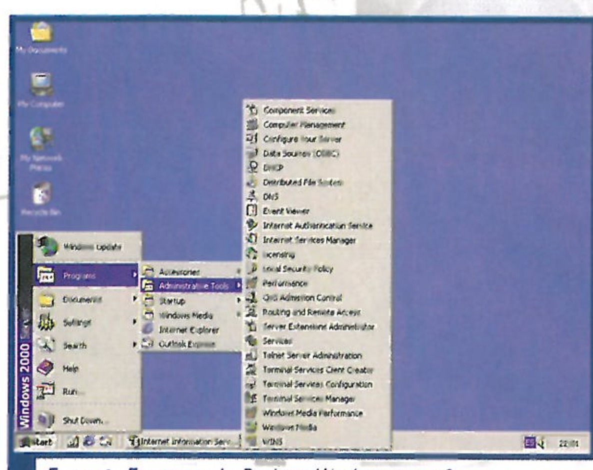
Windows 2000 se encuentra ya en el mercado en sus cuatro versiones: *Professional*, *Server*, *Advanced Server*, y *DataCenter Server* (esta última aún no estaba en el mercado en el momento de escribir el artículo).

En cuanto a *Windows 2000 Professional,* basta decir que está diseñado para establecerse como sistema operativo y cliente de red de gran rendimiento, con una cierta equivalencia con *Windows NT 4 Workstation*. Por ello, tiene los requerimientos de sistema más bajos de las cuatro versiones.

Con *Windows 2000 Server* disponemos de un servidor polivalente para dar servicios a clientes de red, como ficheros, impresoras, *Web*, *DNS*, *WINS*, *Telnet*, *Terminal Server*, etc. *Windows 2000 Server* tiene todas las características de *Windows 2000 Professional*, además de soporte para dos procesadores para nuevas instalaciones, y hasta cuatro en *upgrades* de *Windows NT 4.0*. Ofrece también servicios de *Terminal Server* con los que dar servicio a los ordenadores menos potentes de una red.

Windows 2000 Advanced Server está diseñado sobre la versión *Server*. Añade opciones como el *EMA (Enterprise Memory Architecture)*, y soporte para acceder hasta a 64 *Gb*




Figura 1.- Escritorio de Desktop Windows 2000 Server.

de memoria física en procesadores de 64 *bits*, como los *Alpha* de *Compaq* y los *Pentium* series *Xeon* de *Intel*. Ofrece mayor escalabilidad en cuanto a procesadores en sistemas *SMP (Symmetric Multiprocessing)*. Permite además cuatro procesadores en una instalación nueva, y hasta ocho procesadores, si se parte de un *upgrade* de un *Windows NT 4.0 Enterprise Edition*.

Windows 2000 Server es lo que tendría que haber sido siempre NT

La versión *Advanced Server* soporta *Windows Clustering*, permite combinar dos máquinas con *Windows 2000 Advanced Server* con un conjunto de hasta 64 procesadores en una unidad de administración. Estos servidores actúan como un solo servidor. Esta tecnología ayuda a mantener en pie servicios y aplicaciones en el caso de que un servidor fallase, lo que permite acercarse a una disponibilidad de 24 horas a la semana.

Windows 2000 Data Center Server añade mayor escalabilidad al sistema, al soportar hasta 16 multiprocesadores en sistemas normales, y hasta 32 vía *OEM's* con *hardware* especial. La versión *Data Center* está preparada también para ofrecer servicios a grandes cantidades de datos o procesamiento transaccional. Este sistema está dirigido a empresas con grandes almacenes de datos, o que utilizan procesos transaccionales.

NUEVAS CARACTERÍSTICAS

Windows 2000 viene con cientos de nuevas características, donde existe una base común para las cuatro versiones de *Windows 2000*. En posteriores apartados hablaremos de las nuevas características de *Windows 2000 Server* respecto a *Windows NT 4.0 Server* específicamente.

En la *Web* de *Microsoft* aparece un documento de más de 250 páginas mostrando las nuevas características de su nuevo sistema operativo. A continuación mostramos las más interesantes:

- Rendimiento más rápido. Según *Microsoft*, *Windows 2000* es un 25% más rápido que *Windows 9x* en sistemas con 64 *Mb* de *RAM*. Los usuarios que hayan instalado *Windows 2000* notarán que va más rápido respecto al *NT 4.0 Workstation*. *Microsoft* lo atribuye al mejor uso de la memoria.
- Multitarea más rápida. *Windows* utiliza mejor la arquitectura de 32 *bits*.
- Mayor escalabilidad en cuanto a memoria y procesadores. La versión *Professional* ya llega a 4 *Gb* de *RAM* y dos procesadores. Pocos programas utilizan el doble procesador, resulta más rentable invertir en memoria.
- Se conserva el soporte con versiones anteriores de *Windows 9x* y *NT*. Se puede integrar un *Windows 2000* dentro de un dominio (ver Figura 3) o grupo, y compartir ficheros e impresoras.
- Menús personalizados. El menú de *Inicio* (*Start*) se adapta a la manera de trabajar del usuario, añadiendo los programas que más se utilizan. Es algo similar al menú adaptativo de *Office 2000* y a *Internet Explorer 5*. Se puede deshabilitar en *Settings*.
- Asistentes o *Wizards*. Hay asistentes para todo (*ADS*, usuarios, impresoras, *DNS*, *WINS*.) En la versión *Server* se agradece más que en las otras, ya que muchos administradores han cambiado de sitio o funcionan diferente (ahora todo funciona bajo *MMC*). *Configure your Server* es el *Wizard* por excelencia de la versión *Server*, de tal forma que casi todo se puede configurar a través de este asistente. Aparece la primera vez cuando se arranca *Windows 2000 Server* y se encuentra en el *Administrative Tools*.
- *Troubleshooters*. Son unos pequeños asistentes que ayudan a resolver problemas de configuración y optimización. Se acaba con las interminables ayudas de *Windows*.
- *Find* más rápido. El *Find* o *Buscar* se realiza ahora en un entorno tipo *Web*. Además, en todas las versiones se incluye el servicio *Indexing Service*, que se dedica a indexar todos los datos de los discos duros, trabajando en *background*, de tal forma que se consiguen búsquedas más eficientes. Los usuarios de *Microsoft Office* se acordarán del famoso *Fast Find* que funcionaba de forma similar, aunque era más molesto (por los recursos que requería) que útil. Se puede desactivar pulsando con

el **botón derecho** encima de una unidad o directorio sobre *Properties*. Aún deshabilitando la opción podremos buscar, aunque se tratará de una operación más lenta.

La estructura del ADS de Microsoft es muy diferente a la del NDS de Netware

- Soporte *USB, Universal Serial Bus*. Por fin hay soporte *USB* para un sistema operativo tipo *NT*. Ya se pueden utilizar esos misteriosos conectores traseros que vienen con los nuevos servidores, algo que ayudará a que se generalicen estos periféricos. Al conectar o desconectar un periférico *USB* no hace falta configurar o reiniciar nada.
- Soporte *IrDA*. Añade comunicaciones inalámbricas y seguras entre dos *Windows 2000*, utilizando periféricos compatibles, como los que llevan los portátiles. Se pueden compartir ficheros mediante conexiones *IrDA*.
- *IEEE 1394 (Firewire)*. *Windows 2000* incluye soporte para *Firewire*, aunque la amenaza de la llegada de *USB 2* ha ofuscado su difusión.
- *DVD*. El *DVD* está soportado por *Windows 2000*. El cine y los grandes juegos llegan a *Windows 2000* o al menos eso prometen. Hay soporte *DirectX 7.0*, *OpenGL 1.2*, pero tras probar dos juegos... ¡no van! Para jugar tendremos que esperar al *Millenium*.
- *Plug and Play*. *Windows 2000* permite instalar nuevo *hardware* con pocas complicaciones. *Microsoft* dice que existen 12.000 dispositivos *PNP* que funcionarán con *Windows 2000*. En estaciones de trabajo, el *PNP* está muy bien, aunque en servidores es mejor dejar esta opción deshabilitada.
- Protección de ficheros. Los ficheros del sistema importantes son guardados por *Windows 2000*. Si un programa de instalación trata de sobreescribir uno de estos ficheros clave, saltará *Windows File Protection* y se reemplazará el fichero con su versión correcta. También se protege al ordenador contra el borrado de ficheros de sistema por "accidente".
- Certificación de *drivers*. Los *drivers* pueden venir con un certificado digital, se puede comprobar por *Internet*.
- *Microsoft Installer*. Un servicio que ayuda a los usuarios a instalar, configurar, actualizar y borrar programas correctamente. Así se minimiza el riesgo de error de los usuarios. Los usuarios del *Office 2000* ya habrán podido familiarizarse con esta tecnología.
- Menos *Reboots*. Se han disminuido las circunstancias por las que era necesario reiniciar el servidor. Por ejemplo con algunas instalaciones de *software*.
- *System Preparation Tool (SysPrep)*. Permite clonar configuraciones y sistemas. Resulta muy útil.
- *Setup Manager*. Un *Wizard* que permite crear fácilmente *scripts* de instalación.
- *Remote OS Installation*. Con *SysPrep* o sin él, se pueden instalar máquinas remotamente, disminuyendo el coste de instalación.
- *Multilingual User Interface (MUI)*. Permite cambiar la interfaz de usuario para que pueda leer y escribir documentos en otros lenguajes.
- *Windows NT Security Model*. Los modelos de permisos y auditorías son iguales a los de *NT*.
- Soporte de *Smart Card*. Si se dispone de lectores de *Smart Cards*, se pueden reforzar los *passwords* con tarjetas con *chip*. Hasta que no se inserta la tarjeta, no se puede realizar un *login*.
- Políticas de grupo. La llegada del *ADS* (Directorio Activo) abre nuevas puertas a la administración de usuarios por grupos. Recursos, permisos, aplicaciones, *desktop*, se podrán configurar a nivel de grupo.
- *Windows Management Instrumentation (WMI)*. El *WMI* permite controlar y monitorizar los recursos del sistema con *scripts* y aplicaciones de terceros.
- *Microsoft Management Console (MMC)*. Ofrece una interfaz centralizada a todas las herramientas de administración. Aquí se puede encontrar, por ejemplo las *Disk management tools* (sucesor del *Disk Administrator* de *NT 4.0*). Casi todos los servicios se administran desde una ventana *MMC*.
- *Recovery Console*. Si el sistema se cuelga, *Windows 2000* ofrece la opción de ir a la línea de comandos, donde se pueden parar y encender servicios, leer y escribir ficheros, y algunas otras funciones. No es una ventana *MS-DOS* al completo, pero deja hacer bastantes cosas en caso de apuro.

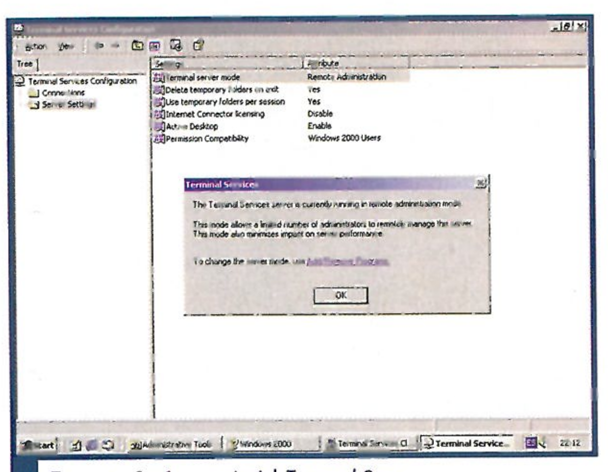


Figura 2.- Configuración del *Terminal Server*.



- *Safe Mode Startup Options*. Al arrancar, se pulsa *F8* y salen las nuevas opciones de arranque. Hay muchas más opciones que antes. Esta solución es más parecida a *Windows 98* (con *Safe Mode*) que a la de *NT*.
- *Microsoft Windows Services for Unix 2.0*. Una serie de servicios orientados a interactuar con sistemas *Unix*.
- *Mobile Wonder*. Siempre habíamos oído afirmar a *Microsoft* que sus versiones de *Windows* están preparadas para los usuarios móviles. Esta vez es verdad, porque *Windows 2000 Professional* es la opción más aconsejable si se dispone de un portátil.
- *IntelliMirror*. En relación con el *Mobile Wonder*, el *IntelliMirror* permite "seguir" los ficheros con los que trabajas, sea cual sea nuestra ubicación en la *Red*.
- *Hard Disk Security*. Se trata de un sistema por el que es posible establecer un *login* y un *password* para acceder al disco duro. Se puede aplicar a las unidades *NTFS* y se convierte en una garantía en caso de robo del equipo (sobre todo si es portátil).
- *Encrypting File System (EFS)*. Encripta cada fichero con una clave aleatoria. El proceso de encriptación y desencriptación es transparente al usuario. Para encriptar o desencriptar un fichero o directorio, basta con acudir a *Properties* con el **botón derecho,** pulsar sobre el botón **Advanced** y elegir las opciones de encriptación. La encriptación añade un nivel adicional de privacidad sobre el ya seguro sistema *NTFS* de *Windows 2000*.
- *Inline Compression*. Ya en *NT 4.0*, los ficheros y directorios se podían comprimir. *Windows 2000* consume menos recursos comprimiendo y descomprimiendo. Es una opción útil sobre todo en portátiles y en servidores, donde el espacio del disco duro siempre acaba siendo insuficiente. Se debería evitar comprimir los ficheros más utilizados.
- Ficheros y directorios *Offline*. Ahora es posible desconectarse de la *Red* y trabajar como si aún se estuviera conectado. Esto permite crear en el ordenador una imagen de los documentos guardados en la *Red*. Simplemente se navega hasta el recurso (ficheros o directorios) en el que se quiere trabajar *Offline*, se pulsa el **botón derecho,** y se marca **Make Available Offline.** De esta manera se iniciará un *wizard* para configurar las opciones.

Todas las herramientas de administración están bajo MMC

Una vez realizado, *Windows 2000* hará trasparente el uso de los ficheros en el disco duro o en la *Red*. Los ficheros de red con propiedades *offline* se identifican con una flecha en la esquina del icono. Los ficheros *Offline* resultan importantes para los usuarios de portátiles, ya que pueden funcionar como una especie de *Backup* automático para ficheros importantes.

- *Smart Battery*. Otra opción para los usuarios de portátiles. Soporta el nuevo sistema de control de batería *ACPI*, así como el viejo *APM*. La batería dura más con *Windows 2000*.
- *Hibernate*. Apaga el ordenador y monitor en un tiempo predeterminado o cuando se cierra su cubierta. El contenido de la *RAM* se vuelca al disco duro antes de que el ordenador se apague. Al reactivar el ordenador, se restaura la *RAM*, y todo queda tal como estaba antes. En la práctica, vemos que cuando no se usa el ordenador es posible cerrar la cubierta, y cuando se tiene que hacer, simplemente se introduce un *password* de seguridad, y todo sigue igual.
- *Hot Docking*. Para conectar un portátil a la *docking station* sin tener que configurar nada.
- *Internet Explorer 5.01*. Totalmente integrado con el *Windows 2000* en todas sus versiones, pero muchas páginas de *Internet* dan algún error de *VBScript*.
- *Search Bar*. Una barra de ayuda donde se pueden buscar diferentes tipos de información, desde *Webs*, *E-mails*. Es posible elegir el buscador que se va a usar.
- *History Bar*. La barra de historial que viene con *IE 5.01* es omnipresente. No sólo guarda *URL's*, sino servidores de red, *URL's* de *Intranet*, directorios locales...
- *AutoComplete*. Al escribir una *URL* o recurso de red, se completa el nombre. Parecido a *IE 4.0* o *Netscape*, pero con mucha más memoria.
- *Internet Explorer Administration Kit (IEAK)*. Permite a los administradores de red distribuir en

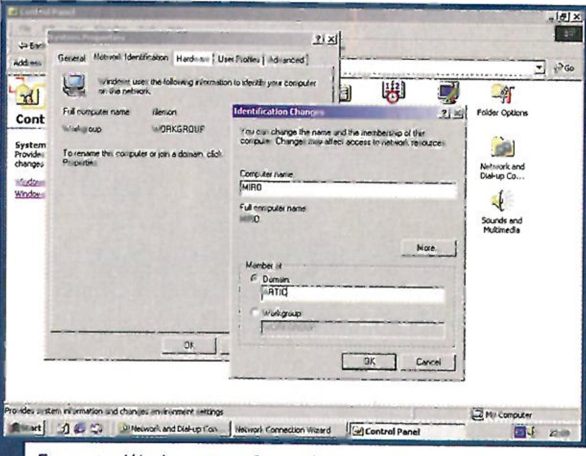


Figura 3.- *Windows 2000 Server* dentro un dominio *NT*.

las estaciones de trabajo *Internet Explorer 5.01* con los componentes que se elijan. Todo desde un puesto de trabajo.

- *AutoCorrect*. Automáticamente corrige las faltas al escribir *URL's*.
- *Automated Proxy*. *IE 5.01* intenta localizar un *proxy* automáticamente para conectarse a la *Red*.
- *Internet Connection Sharing*. *Windows 2000,* en todas sus versiones incluye *Network Translation Address (NAT)*. Con sólo configurar una conexión a *Internet* vía módem, por ejemplo, toda la red se podrá conectar a *Internet* a través del ordenador de la conexión.
- *NetMeeting*. El *NetMeeting* ya es una parte más de *Windows 2000*. Permite videoconferencia, intercambio de ficheros, *chat* entre dos, etc...
- *Offline Viewing*. Se pueden ver páginas *Web* enteras con gráficos cuando no se está conectado. Útil para ordenadores portátiles cuando no pueden efectuar una conexión.
- *Wizards* para acceso remoto. Ahora es mucho más fácil conectar a redes remotas o redes tipo *VPN*.

REQUERIMIENTOS

Los requerimientos mínimos que indica *Microsoft* son: un *Pentium* 133 *Mhz* con 64 *Mb* de *RAM* y un disco de 1 *Gb*. En una máquina de esas características con la instalación de *Windows 2000 Professional* para pruebas, no funciona mal, pero la citada versión recién instalada, ya ocupa 650 *Mb* y no deja sitio para mucho más.

En la citada máquina, al instalar la versión *Server* de *Windows 2000,* arranca bien en principio, pero al iniciar servicios la máquina se hunde completamente. Hay que recordar, que *Windows NT 4 Workstation* era una versión de *NT Server* recortada. En este caso, *Windows 2000 Server* es mucho más que la versión *Professional*, con infinidad de servicios añadidos. Una vez instalado *Windows 2000 Server* con las opciones por defecto ocupa unos *700 Mb*.

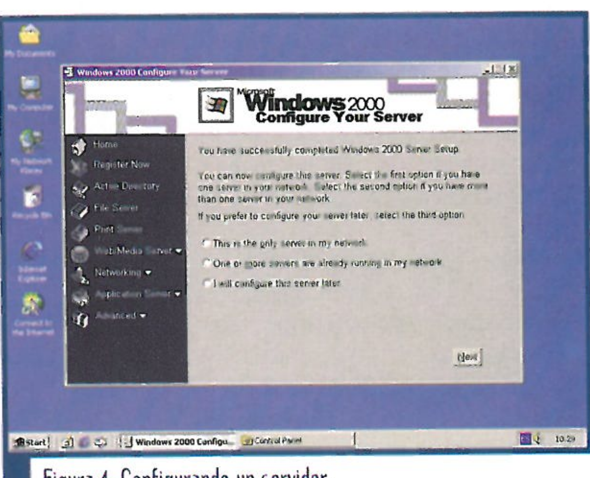


Figura 4.-Configurando un servidor.

Migrar de un dominio NT a ADS requiere mucha planificación

Si efectuamos una prueba, con una máquina más acorde con los servidores de hoy en día, como es un doble procesador *Pentium II* 350 *Mhz* con 256 *Mb* de *RAM*. Dicha máquina dispone también de una tarjeta *Mylex* tipo *DAC960* para *RAID* por *hardware*. Los discos son dos *Seagate UWSCSI* puestos en *RAID* 0. En esta configuración, *Windows 2000 Server* funciona de manera fluida, algo mejor que cuando actúa como servidor de pruebas para *NT 4*.

INSTALACIÓN

Para iniciar la instalación se puede arrancar desde el mismo *CD-ROM*, evitando así los disquetes de instalación. La primera fase es igual que en el caso de *NT 4*: crear particiones, seleccionar una, formatear y copiar ficheros.

Se agradece mucho el soporte de *FAT32*, ya que permite crear particiones grandes que después se pueden convertir o no a *NTFS*. Después de la copia de ficheros, reiniciaremos el ordenador.

Comienza así, la parte gráfica al más puro estilo de la instalación de *Windows 98*. Recuerda a *Windows 98* por el entorno de instalación, y por el tipo de preguntas efectuadas, ya que apenas aparecen preguntas técnicas durante la instalación. Hay que entender un poco cuando se escogen los servicios a instalar, un proceso fácil para aquellos acostumbrados a instalar *Windows NT 4*.

Todo se configura después mediante *wizards* en el primer inicio de *Windows 2000 Server* (y en los posteriores). Como mucho, hay una pregunta referente al modo de optimización del *Terminal Server*, pero tampoco implica dificultad. De esta forma, cualquiera que sepa instalar *Windows 98* conseguirá hacer lo propio con *Windows 2000 Server*. Al soportar *Plug and Play*, *Windows 2000 Server* no realiza tampoco pregunta alguna referente a la configuración del *hardware*.

Hay muchos *drivers* nuevos incluidos, sobre todo, en tarjetas



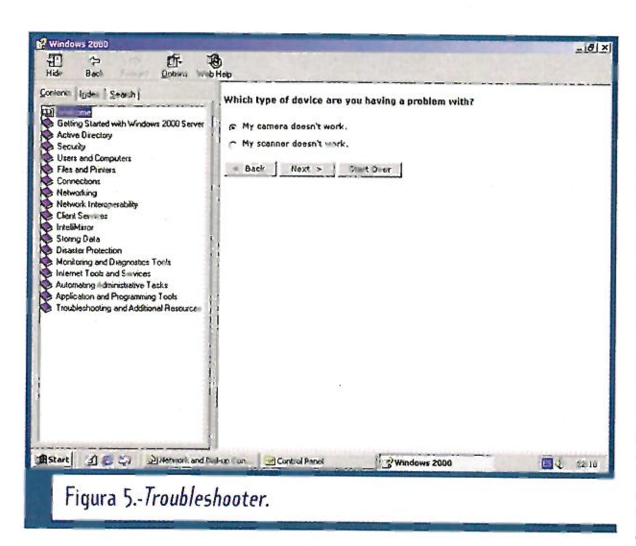


Figura 5.-*Troubleshooter.*

para dispositivos de almacenamiento tipo *SCSI*. Cada vez que se tenía que instalar *Windows NT 4* en una máquina relativamente nueva, era más alta la montaña de disquetes con *drivers* adicionales que el servidor.

En la prueba realizada con el *Pentium II*, *Windows 2000 Server* reconoció la tarjeta *Mylex* de *RAID*, e incluso la *Voodoo Banshee* (es un servidor especial). Pero no detectó la tarjeta de red, una *Kingston* con *chipset Digital*. Una tarjeta detectada generalmente como *Tulip*, que ya se detectaba en *NT 4.0* y que es reconocida por todos los sistemas operativos de tipo servidor del mercado (*Linux*, *SCO*, *Netware*).

Si se revisa la lista de tarjetas de red soportadas, se aprecia que la lista de *Digital* sólo incluye una entrada, una tipo *ISA*, y que la lista de *Compaq* está sin actualizar.

Todos los dispositivos no detectados requieren un paseo por la *Red* en busca de *drivers*. Las grandes marcas ya los tienen disponibles, algunas con el nombre de *NT 5.0*. Al utilizar *hardware* de fabricantes pequeños hay dificultades para encontrar *drivers*.

Decididamente, las llegadas de la *FAT32* y el *PNP* facilitarán bastante las cosas a los técnicos de sistemas que vayan a instalar nuevos servidores.

Una vez acabada la instalación, el sistema operativo reiniciará y arrancará *Windows 2000 Server* por primera vez. Si algo ha ido mal, aparecerá un error que indica que se ha producido al menos un error al arrancar, entonces acudiremos al *Event Manager* (ver Figura 9) para obtener una información más detallada.

La premisa número uno de diseño del nuevo Windows fue la estabilidad

Al arrancar *Windows 2000 Server* pedirá un *login* y una clave de acceso, tras lo que aparecerá el *super-Wizard*: *Configure your Server*. Desde este asistente será posible configurar casi todos los servicios y parámetros de nuestro servidor.

La primera vez que se instala *Windows 2000 Server*, el citado asistente resulta de gran ayuda, ya que se han cambiado de lugar o de nombre algunas opciones y el *wizard* permite aprender los nuevos conceptos y la nueva ubicación de algunas herramientas.

Se puede insertar un servidor en un dominio existente *NT* o grupo. En el *Server Manager* de *NT 4* aparece como *Servidor NT 5.0*. También se puede iniciar *ADS* (*Active Directory Services*). Del *ADS* se han escrito libros enteros. Los que hayan trabajado con *Netware 4.0* o posterior entenderán rápidamente *ADS*, ya que es muy similar al *NDS* de *Netware*.

El *ADS* es otra manera de organizar los usuarios y servidores diferente a la del dominio *NT*. En un dominio *NT*, la base de datos de usuarios es plana, sin embargo en *ADS* todo está organizado en forma de árbol. En un dominio *NT* no puede haber usuarios con un mismo nombre, mientras que en *ADS* un usuario está identificado por su nombre de usuario y su contexto (contexto es la situación dentro del árbol). Un mismo nombre de usuario puede estar en dos ramas del árbol a la vez (por ejemplo *Marketing* y *Finanzas*).

Tras la instalación y configuración, podemos empezar a trabajar con la administración. Ésta, se realiza desde las herramientas situadas en **Start->Programs->Administrative Tools**, al igual que bajo *NT 4*. También se puede llegar desde el *Control Panel* o a través del asistente *Configure your Server*.

Todas las herramientas tienen aspecto de navegador, realizándose toda la administración desde ventanas tipo *MMC*, que ayudan a tenerlo todo organizado y fácil de entender. El aspecto gráfico es atractivo, incluso para ser un servidor.

No hay que olvidar el hecho de mover una ventana rápidamente por la pantalla que puede llegar a coger un 35-40% de *CPU* en ambos procesadores. Por suerte, los difuminados y animaciones se pueden deshabilitar. El entorno es parecido a *Windows 98*, pero con las funcionalidades de un *Windows NT*. Se podría afirmar que *Windows 2000 Server* es lo que *NT* siempre tendría que haber sido.

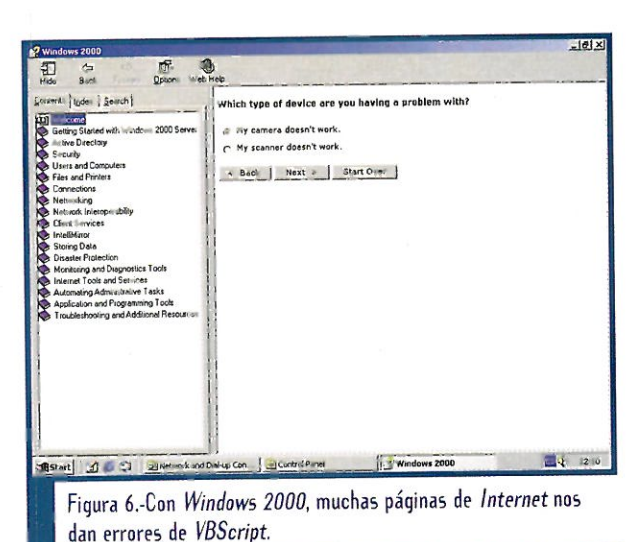


Figura 6.-Con *Windows 2000*, muchas páginas de *Internet* nos dan errores de *VBScript*.

un fichero de *Release* que cuenta los errores conocidos. Además, las *news* están llenas de problemas con *drivers*, sobre todo productos *USB* y tarjetas de *Red*. Tendremos que esperar unos meses para ofrecer un buen diagnóstico.

¿VALE LA PENA ACTUALIZARSE?

Para contestar a esta pregunta, habría primero que comparar *Windows NT 4 Server* con *Windows 2000 Server*. Hay que tener en cuenta la afirmación de *Microsoft*, cuando dice que uno de los principales parámetros que tuvo en cuenta al diseñar este nuevo sistema operativo fue la estabilidad.

Todo administrador ha sufrido la falta de estabilidad de *Windows NT 4.0 Server*, circunstancia que ha evitado que triunfase como servidor de *Internet*. Los técnicos volvieron al mundo *Unix* como plataforma estable y más ahora con el fenómeno *Linux* rondando por *Internet* y los medios de comunicación.

Debido a la juventud de este sistema operativo y a la falta de aplicaciones que saquen jugo a sus posibilidades, no podemos decidir si es más estable o no que *NT 4*. Hay que tener en cuenta que *Windows 2000 Server* aún no había salido al mercado y *Microsoft* ya publicó un *patch* para *Indexing Service*.

En el directorio raíz del *CD-ROM* de instalación encontraréis

Un técnico querrá saber si su red va a funcionar mejor, más rápido, y con una administración más simple si se actualiza a *Windows 2000 Server*. La actualización a *Windows 2000 Server* no implica una procedencia de *Windows NT Server*, ya que el mercado de los sistemas operativos de servidor no está tan copado como los de sobremesa.

Al igual que en NT, pocos juegos funcionarán bajo Windows 2000

Muchas de las novedades que incorpora *Windows 2000 Server* ya existían en *Netware 4.0* y más tarde en la *5.0*. Recogemos a continuación algunas de las principales novedades.

SERVIDOR DE FICHEROS E IMPRESIÓN

Ninguna gran consultoría ha sacado todavía ningún análisis definitivo sobre el rendimiento de *Windows 2000 Server* como servidor de ficheros e impresión. En máquinas grandes con multiprocesador, *Windows 2000 Server* tendría que ser el líder frente a *NT* y otros competidores como *Unix* o *Netware*.

Esto se debe a que es el primer sistema operativo capaz de dividir el trabajo de ficheros e impresoras entre procesadores.

RENDIMIENTO

Microsoft afirma que *Windows 2000 Server* obtiene mejores rendimientos del *RAID* por *software* que *NT 4* y la competencia. Para servidores en producción con datos críticos, resulta más conveniente utilizar soluciones *RAID hardware*.

Microsoft también ha añadido su *Hierarchial Storage Management* (*HSM*) para controlar todos los almacenes de datos en una único punto. El *HSM*, es como una evolución del *NFS*, o productos parecidos que existían ya en *Netware* y *UNIX*. Sólo es aplicable a grandes empresas, que lo más seguro es que si lo necesitaban, ya lo tengan. Según *Microsoft*, el rendimiento sirviendo ficheros aumenta un 7% actualizando a *Windows 2000 Server*.

En cuanto a administración se refiere, *Windows 2000 Server* ha mejorado bastante. Incorpora nuevas mejoras en cuanto a ficheros y discos, como son las cuotas de disco, soporte para unidades removibles y el sistema *HSM*.

Estas nuevas facultades pueden ser suficientes para algunos, y dar pie a la actualización. Con relación a las impresoras, *NT* y *2000 Server* ofrecen más o menos las mismas características. Utilizando *ADS* puedes hacer búsquedas más rápidas de impresoras según características (láser, color), o buscar la que está más cerca de un determinado sitio, ya que las impresoras cuelgan del árbol *ADS*.

HERRAMIENTAS DE ADMINISTRACIÓN

En los mundos *Netware* y *Unix* aún nos encontramos con administra-

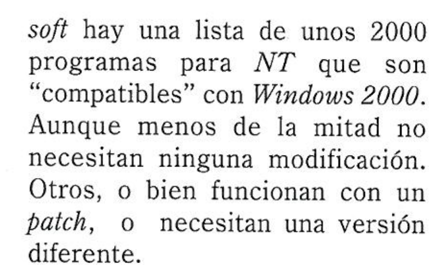


ción bajo línea de comandos, aunque ya van migrando hacia entornos *Web* o gráficos.

Aun no hay análisis definitivos de rendimiento del nuevo Windows 2000 Server

Windows NT 4 también tiene una herramienta distinta para cada recurso (usuarios, servidores, *Web*), pero con las últimas versiones de *BackOffice Server*, ya empezaban a salir los productos con *Microsoft Management Console* (*MMC*). En *Windows 2000 Server* toda la administración se realiza bajo *MMC*, éste entorno lo que permite la familiaridad con la administración de diferentes recursos.

Algunas operaciones, como mover usuarios entre dominios se hacen de manera más fácil. Sin embargo, el *MMC* no representa ningún gran avance para los administradores avanzados. No aporta muchas soluciones a la hora de administrar una gran empresa con muchos usuarios y servidores. Además, las herramientas para tratamiento de usuarios masivos tipo *Batch* son escasas.



Con la llegada del *ADS, Microsoft* prometía un gran avance en la administración de estaciones de trabajo. El problema llega, cuando las estaciones tienen que ser *Windows 2000 Professional*. Pero, por el momento, sólo podrán beneficiarse del *ADS* las estaciones y servidores *Windows* 2000.

Windows 2000 Server destaca cuando hay que añadir o cambiar *hardware* en el servidor, gracias a la llegada del *PNP* y la inclusión de gran variedad de *drivers* que facilitan mucho el proceso.

SERVIDOR DE APLICACIONES

Como servidor de aplicaciones, *Windows NT* tiene muchas más aplicaciones que cualquier otro sistema operativo, ya sea *Netware* o *Unix*. Más adelante saldrán versiones que utilicen las nuevas características de *Windows 2000 Server* y entonces será el momento de actualizar. Cabe decir que si vuestro servidor de aplicaciones se ha quedado pequeño, *Windows 2000 Server* permite hasta 4 *Gb* de *RAM*, el *Advanced Server* 8 Gb, y el próximo *Data Center*, hasta 64 *Gb*.

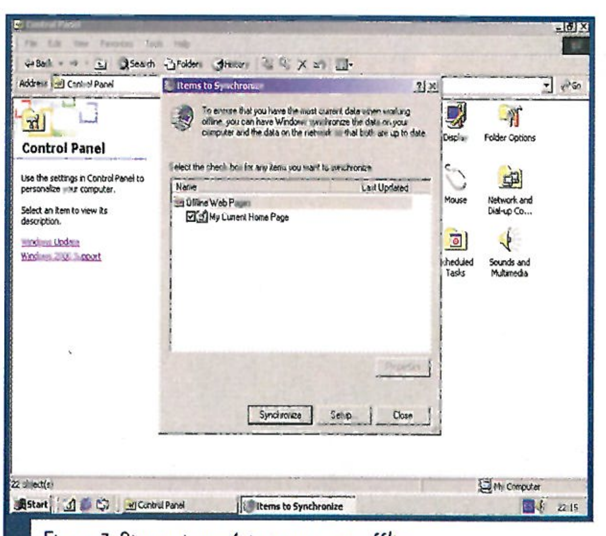


Figura 7.-Sincronizar páginas para ver *offline*.

Quizás, por razones de ampliación, sí puede ser interesante la actualización. No hay que olvidar que todas las aplicaciones que funcionan bajo *NT*, como servidores de correo o bases de datos, funcionan bajo *Windows 2000 Server*. De hecho, en la *Web* de *Microsoft* hay una lista de unos 2000 programas para *NT* que son "compatibles" con *Windows 2000*. Aunque menos de la mitad no necesitan ninguna modificación. Otros, o bien funcionan con un *patch*, o necesitan una versión diferente.

En breve, el mercado estará inundado de aplicaciones para *Windows 2000*. Por el momento, algunas utilidades de *Microsoft*, como el *Exchange 5.5* pueden interactuar con *ADS*.

SEGURIDAD

NT se ve muy superado en materia de seguridad por *Windows 2000 Server*, ya que este último ofrece soporte de *SmartCards*, encriptación *IPSec*, autenticación por *Kerberos*, la inclusión de un servidor *Radius* y uno de certificados, además de la seguridad que lleva en sí el sistema de *ADS*.

No hay muchas razones convincentes para actualizar los servidores

Estas características funcionan bien para las grandes empresas, donde la seguridad y la gestión son muy importantes. Una pequeña empresa no le sacaría tanto provecho respecto a *NT*, ya que la seguridad de *Windows 2000 Server* está pensada para corporaciones.

SOPORTE INTERNET

Windows 2000 Server ofrece más o menos lo mismo que *NT*, *Netware* o *Unix* en cuanto a servicios para *Internet*. Pero, incluye el nuevo *Internet Information Server 5.0* (*IIS 5.0*), cuya principal ventaja respecto el *IIS 4.0* es que puede manejar

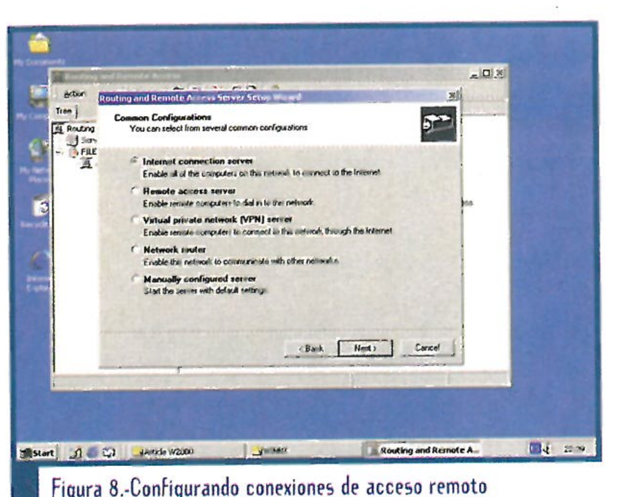


Figura 8.-Configurando conexiones de acceso remoto

múltiples *sites* en una misma máquina de manera más eficiente.

Microsoft ha combinado las prestaciones del servicio *DNS* y *ADS*. Muchos *DNS* no corren sobre *Windows,* sino sobre *UNIX*. Con ello, *Microsoft* afirma que es compatible con la versión 8.1 de *BIND*. Sólo las grandes empresas podrán beneficiarse de una actualización a *Windows 2000 Server*.

UTILIDADES DE CONVERSIÓN

Quizá los usuarios crean que con la llegada de *ADS*, ahora *Netware 4* y *5* son más fácilmente exportables. Bueno, ahora simplemente es posible hacerlo en buenas condiciones, aunque la filosofía de los árboles es muy diferente.

Muchos se pasaron a UNIX (Linux) por razones de estabilidad

Como los dos árboles se basan en *LDAP3*, no tardarán mucho en aparecer herramientas de terceros o de la propia *Microsoft*. Se pueden actualizar *NT's* que no sean controladores de dominio a *Windows 2000 Server* con sólo instalar encima, integrándolos dentro del dominio *NT*.

El problema viene al querer pasar del dominio *NT* a *ADS*. Requiere mucha planificación, sobre todo en empresas grandes. *Microsoft* ofrece herramientas para realizar esta migración.

Para empresas pequeñas, no vale la pena pasarse a *ADS*. Tampoco creo que valga la pena de momento migrar los servidores que no son controladores de dominio *NT* a *Windows 2000 Server*. Desde luego, las empresas grandes pueden beneficiarse de *ADS*.

PROPIEDADES DE UNA CLASE

Tanto *Windows* 2000 como *NT* siempre han requerido más *CPU* y memoria que *Unix* o *Netware*. *Windows 2000* presenta diferentes ventajas respecto a *NT*: utiliza mejor el *SMP*. Pero también desventajas como utilizar más memoria y más espacio en disco que *NT*, de 180 *Mb* el *NT* a 700 *Mb* la versión 2000.

Con un *Pentium* II a 350 *Mhz*, *Windows 2000* empieza a sentirse cómodo. La *RAM* será la principal razón por la que muchas máquinas se queden sin actualización. *Windows 2000 Server* recomienda 128 *Mb*.

Como en el caso de todos los sistemas operativos nuevos, *Windows 2000 Server* necesitará un poco de tiempo antes de que se considere probado.

CONCLUSIÓN

Hay casos concretos en los que se podría aceptar la actualización. *Microsoft* dará un gran paso el día en que lance una versión de *Back Office* que saque el jugo a *Windows 2000 Server*.

Podemos estar seguros de que en breve empezarán a salir las nuevas máquinas ya con *Windows 2000 Professional* instalado. El número de *drivers* y dispositivos compatibles irán en aumento. Todas las aplicaciones de servidor que salgan, ya estarán certificadas para *Windows 2000 Server,*

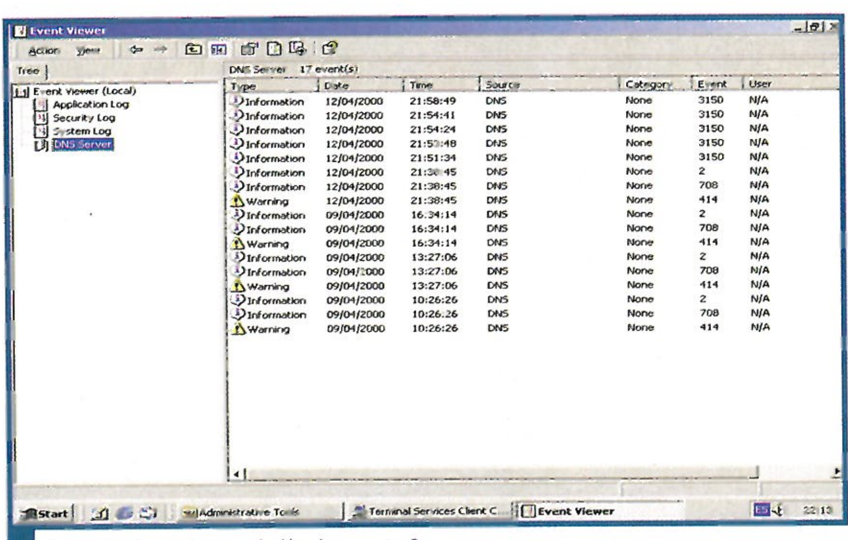


Figura 9.-*Event Manager* de *Windows 2000 Server.*


e-business
Hace falta un software muy potente para convertir una estrategia para Internet en un negocio en Internet en 60 días.
Así es el software de IBM.
IBM
WebSphere Application Server y Lotus Domino son los servidores de aplicación que le pueden ayudar a desarrollar nuevas aplicaciones Web en semanas, no en meses. Tanto si está creando una empresa ".com" como compitiendo con una de ellas, es una ventaja decisiva. Descubra qué pueden hacer por su negocio en www.ibm.com/software/soul/build/es
El software es la energía de e-business



Curso de Programación con WAP (II)

Constantino Sánchez Ballesteros.
Técnico Superior Desarrollo Aplicaciones Informáticas.

Al igual que en las páginas *Web* que vemos en nuestros *PC's* tenemos la posibilidad de visualizar imágenes mediante *WML* (entre otras características multimedia), con el fin de que nuestras páginas *WAP* sean más ricas en contenido y más vistosas.

LIMITACIÓN DEL HARDWARE

Dado que *WAP* es una tecnología que acaba de nacer, las limitaciones que aún tiene son relevantes. La transferencia de datos no es ni mucho menos una panacea y el *hardware* utilizado en los móviles no permite demasiado margen para la implementación de páginas interactivas ricas en detalles.

De momento, para insertar imágenes en nuestras páginas *WML* tendremos que crearlas con programas de conversión gráfica específicos y a un formato determinado. Dado que en la *Red* podemos encontrar multitud de conversores al formato de imagen utilizado por *WML*, nos basaremos en uno en particular: **PIC2WBMP**.

PIC2WBMP

Las imágenes utilizadas en *WML* deben estar en formato *wbmp*. Este formato especial utiliza imágenes en dos colores, y dependiendo del móvil que visualice la imagen, debemos tener en cuenta cuál debe ser su tamaño adecuado.

Este conversor permite importar imágenes en los formatos más comúnmente usados (*JPEG, GIF,* etc.). Cuando se importa una imagen, tenemos en el formulario del programa dos vistas (Figura 1).

Como se puede apreciar en la Figura 1, cuando se convierte una imagen con más de dos colores se pierde resolución, ya que en la conversión se pasará la imagen original a un formato de tan sólo dos colores.

Este hecho obliga a poner cuidado en la selección de imágenes a convertir si no queremos estar retocándolas posteriormente con frecuencia. Cuando en próximas


e-business
Hace falta un software muy potente para que lo que desarrolle hoy sirva mañana, el mes que viene o el próximo año.
Así es el software de IBM.
IBM
Java, XML, Linux... está claro que sin soporte es imposible estar y mantenerse al día. El software de IBM y los recursos que ponemos a disposición de los desarrolladores le ayudarán a estar siempre por delante. Descubra cómo en www.ibm.com/developerworks
El software es la energía de e-business

TABLA 1

ATRIBUTO	DESCRIPCIÓN
alt=vdata	Especifica una representación textual alternativa para la imagen utilizada cuando ésta no pueda ser visualizada por cualquier otro método, es decir, que la imagen no se pueda encontrar o el móvil no la pueda visualizar correctamente.
src=ref.	Especifica la *URL* donde se encuentra la imagen. El navegador la descargará desde la *URL* indicada y la visualizará cuando se esté representando el texto.
Localsrc=vdata	Especifica una representación interna alternativa para la imagen.
vspace=length hspace=length	Estos atributos especifican la cantidad de espacio en blanco que se insertará de izquierda a derecha *(hspace)* y de arriba a abajo *(vspace)* sobre la imagen u objeto.
align= (top|middle|bottom)	Especifica el alineamiento de la imagen junto con el texto respecto al punto actual de inserción. Este alineamiento tiene tres valores posibles: - *Bottom*: el pie de la imagen se alinea verticalmente con la línea base actual. Es el valor por defecto. - *Middle*: el centro de la imagen se alinea verticalmente con el centro de la línea de texto actual. - *Top*: la cabecera de la imagen se alinea verticalmente con la cabecera de la línea de texto actual.
height=length width=length	Estos atributos especifican el tamaño de una imagen u objeto. Los móviles pueden escalar las imágenes u objetos para que se correspondan con los valores asignados.

TABLA 2

ATRIBUTO	DESCRIPCIÓN
Title=vdata	Especifica una breve cadena de caracteres identificando el enlace. Para que este elemento trabaje correctamente con un amplio rango de móviles debemos mantener nuestras etiquetas por debajo de seis caracteres.

revisiones de *WML* (y por supuesto el *hardware* utilizado para los móviles) soporten imágenes con más colores, podremos hacer maravillas, pero mientras hay que conformarse con lo que hay.

Los elementos se utilizan constantemente para insertar imágenes o texto formateado

Para convertir una imagen en el programa *PIC2WBMP* tan sólo tendremos que utilizar el menú *File/Open* y seleccionar la imagen que deseamos utilizar. Una vez cargada en el programa se mostrará automáticamente la imagen original y la que quedaría después de convertirla.

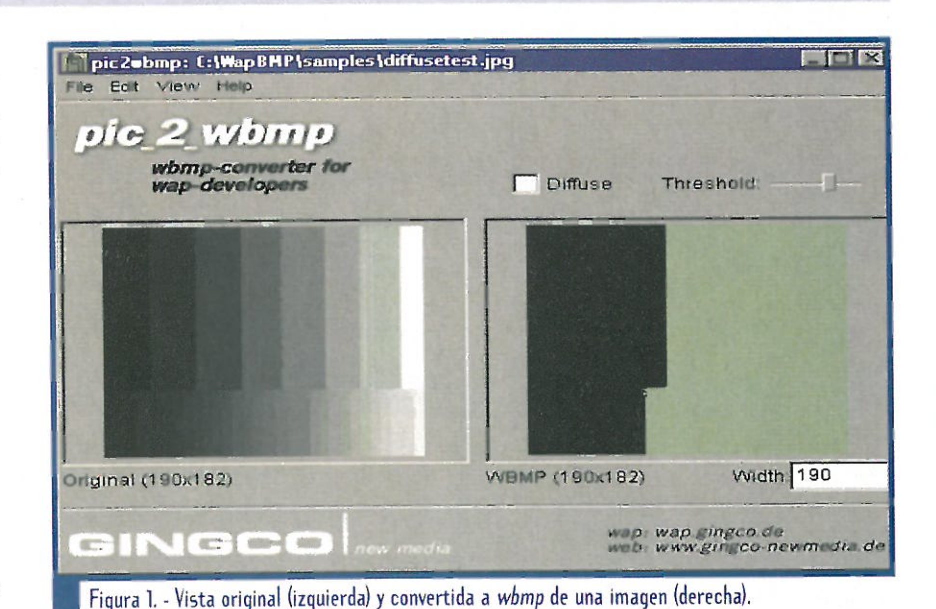


Figura 1. - Vista original (izquierda) y convertida a *wbmp* de una imagen (derecha).

Hay que hacer constar el hecho de que este programa funciona bajo *Java* y por lo tanto, debemos tener el *Runtime* en nuestro equipo para que se pueda ejecutar.

Para finalizar, si nos ha gustado el aspecto final de la conversión sólo tenemos que ir al menú *File/Save WBMP* y guardar nuestra imagen en un directorio para su uso posterior.



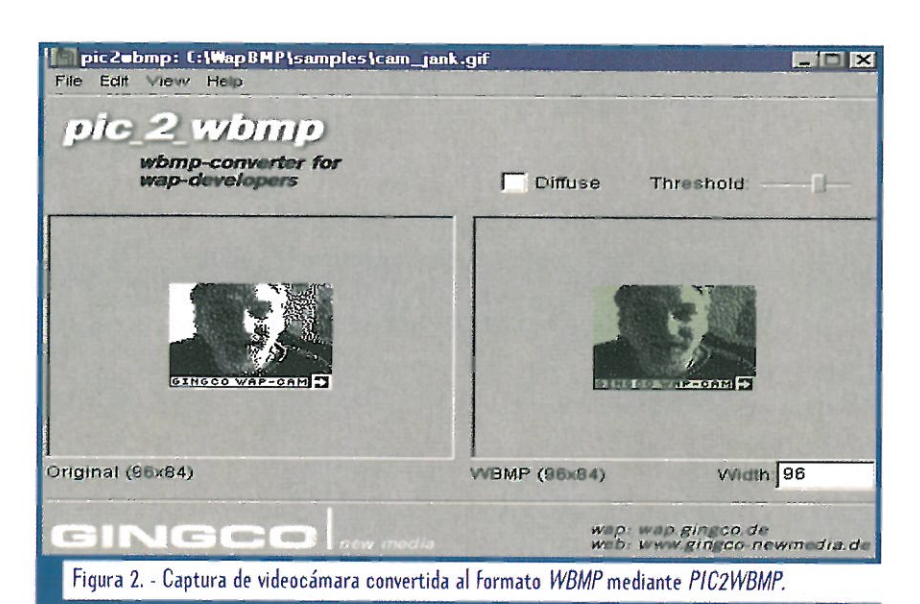


Figura 2. - Captura de videocámara convertida al formato *WBMP* mediante *PIC2WBMP*.

Se ha insertado un espacio en blanco de izquierda a derecha *(hspace)* así como de arriba a abajo de la imagen *(vspace)*.

ELEMENTO IMG

El elemento *img* lo utilizaremos dentro de nuestros programas *WML* para visualizar una imagen en el móvil. Las imágenes utilizan el mismo formato que el texto normal y podemos encontrar su sintaxis en la Tabla 1.

En el siguiente ejemplo podemos ver el uso del elemento *img:*

```
<img src="imagenes/grafico.wbmp"
    alt="Mi gráfico"
    src="default.wbmp" hspace="1"
    vspace="1"/>
```

En este caso, *src="imagenes/grafico.wbmp"* se refiere al fichero *grafico.wbm* que está localizado en el directorio *imagenes* dentro del mismo dominio utilizado por el *deck* que utiliza el elemento.

ELEMENTO ANCHOR

Este elemento especifica la cabecera de un enlace *(link)*. Podemos utilizar *anchor* en cualquier lugar donde el texto formateado se pueda poner, excepto para elementos *option*. Un enlace que utilice *anchor* debe tener asociado una tarea expresa que especifique un comportamiento cuando el *anchor* se seleccione.

Un *anchor* puede contener los siguientes elementos:

- *go*
- *prev*
- *refresh*
- *br*
- *img*

Los atributos del elemento *anchor* se detallan en la Tabla 2.

En el siguiente ejemplo se muestra la utilización del elemento *anchor:*

```
<wml>
<card id="enlaces" title=
    "Enlaces">
<p>
Esto es un texto normal, y aquí
    hay un
<anchor title="ENLACE">Enlace!
<go href="files/archivo.wml">
<setvar name="var_name"
    value="var_value"/>
</go>
</anchor>
</p>
</card>
</wml>
```

LISTADO 1

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="card1" title="Ejemplo 1">
    <p>
  <img src="jeto.wbmp" alt="Cara"/>
  <anchor title="Link"> Elemento anchor
  </anchor>
  <a href="destino.wml"> Elemento a</a>
    </p>
  </card>
</wml>
```

TABLA 3

ATRIBUTO	DESCRIPCIÓN
Title=vdata	Especifica una breve cadena de caracteres identificando el enlace.
Ref.	Especifica el *URI* de destino, por ejemplo, el *URI* del *card* que visualizar.

TABLA 4

TIPOS DE DATOS	DESCRIPCIÓN
CDATA	Texto que puede contener caracteres numéricos o alfanuméricos. Sólo puede utilizarse para valores de atributos.
NMTOKEN	Un nombre *token*, contiene cualquier combinación de números, letras, y algunos caracteres de puntuación, a destacar ".", "-", "_", y ":". Podemos iniciar un *token* con un carácter de puntuación. Un token no puede utilizarse como referencia a una variable o nombres de elemento.
Id	Es un identificador único para un elemento determinado.

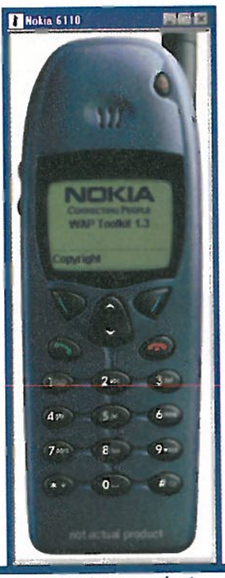


Figura 3. - *NOKIA* 6110 incluido en el entorno de desarrollo de *Nokia*.

ELEMENTO A

El elemento *A* es un formato corto del elemento *anchor* y permite ejecutar una tarea sin utilizar variables.

Comparemos código. El siguiente utiliza el elemento *anchor:*

```
<anchor>Púlsame
<go href="destino.wml"/>
</anchor>
```

CUADRO 1

Nombre	Tipo	Descripción
%text	#PCDATA	Indica que el texto está formateado

Este código se puede escribir produciendo el mismo resultado con el elemento *A*:

```
<a href="destino.wml">
    Púlsame</a>
```

En la Tabla 3 podemos ver los atributos del elemento *A*.

En el Listado 1 podemos ver un ejemplo codificado que utiliza los elementos vistos anteriormente.

TIPOS DE DATOS EN WML

En la Tabla 4 se explican diferentes tipos de datos que podemos utilizar o encontrarnos en páginas *WML*.

Ejemplo de *CDATA:*

```
"$(valor)"
nombre="valor"
```

Ejemplo de *NMTOKEN:*

```
"text"
_card1
a.name.token
.un-token-.correcto
```

Ejemplo de un identificador *(id):*

```
<card id="card1"/>
```

OTROS TIPOS DE DATOS DE INTERÉS

LENGTH

Puede especificarse como un entero y representa el número de píxeles de la pantalla o un porcentaje del espacio horizontal o vertical disponible. Por ejemplo, el valor "50" significa 50 píxeles. Para el ancho, el valor "50%" significa la mitad del espacio horizontal disponible.

El valor entero se forma con uno o más dígitos decimales ([0-9]) seguidos por un carácter de porcentaje opcional (%). El tipo *%length* sólo se utiliza en valores de atributos. Veamos un ejemplo:

Los valores que contienen los atributos *hspace* y *vspace* son del tipo *%length*.

```
<img src="bitmaps/moon.wbmp"
     hspace="1" vspace="1"/>
<img src="bitmaps/moon.wbmp"
     hspace="2%" vspace="2%"/>
```

VDATA

Representa una cadena que puede contener referencias a variables. Este tipo, al igual que el anterior, sólo se utiliza en valores de atributos.

Ejemplo:

```
<card id="card1" title="$(acti
```

FLOW, INLINE Y LAYOUT

El tipo *%flow* representa el nivel de *card* y *%inline* información sobre el nivel del texto. En general, *%flow* se puede incluir en cualquier marcador. El tipo *%inline* indica áreas donde sólo se gestiona el texto puro o las referencias a variables.

TEXT

Puede incluir las entidades del Cuadro 1.

Veamos un ejemplo de línea con texto normal.

```
<em>
<strong>
```

En el caso de una línea de texto resaltado:

```
</strong>
</em>
```

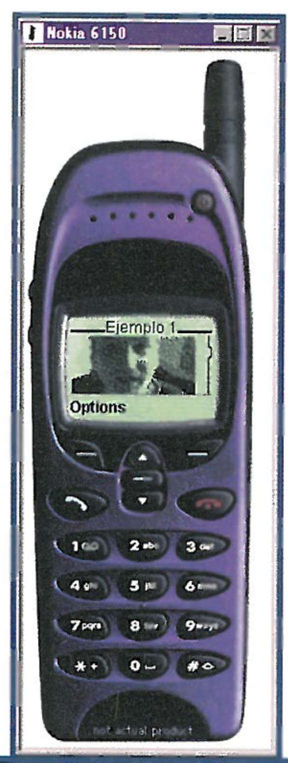


Figura 4. - Nuestro primer ejemplo en ejecución.

HREF

Se refiere a un *URI* (Identificador de recurso uniforme) relativo o absoluto. Puede contener referencias a variables.

```
<go href="http://wapforum.org/"/>
<go
    href="file:///d:\dir\pagina.w
    ml"/>
<go href="aplicacion.wml"/>
```

Los valores de los eventos son *URL's:*

```
<card onenterforward="#card2"/>
```

El atributo *src* del elemento *img* será en esta ocasión una *URL* como la siguiente:

```
<img src="earth.wbmp"/>
```

BOOLEAN

Representa un valor lógico (verdadero o falso).

Ejemplo:

```
<card newcontext="true"/>
<do optional="true"
    type="accept"/>
```

NUMBER

Representa un valor entero mayor o igual que cero.

Ejemplo:

```
<select tabindex="2"/>
<input name="setvar" size="4"
    maxlength="20" tabindex="3"/>
```

ÉNFASIS (EMPH)

El tipo *%emph* cubre los *tags* de texto formateado.

Ejemplo:

```
<em>
```

Línea con énfasis:

```
<big>
```

Una gran línea con énfasis:

```
</big>
</em>
```

CUADRO 2.

Entidad	Notación	Descripción
quot	"	comillas
amp	&	ampersand
apos	'	apóstrofe
lt	<	menor que
gt	>	mayor que
et	=	igual que

Nota: El punto y coma (;) forma parte de la secuencia de escape de un carácter especial.

CARACTERES Y CARACTERES ESPECIALES

Un caracter codificado determinado puede que no sea capaz de expresar todos los caracteres que hay en un documento. Cuando en algún momento no podamos introducir caracteres al documento de forma directa, podemos utilizar las "entidades caracter".

Las entidades caracter son independientes del mecanismo utilizado para introducir un caracter determinado. *WML* soporta entidades caracter numéricas y alfanuméricas.

WML soporta los siguientes formatos de entidades caracter:

- Alfanuméricos, como *&amp e &lt*
- Decimales, como &#123
- Hexadecimales como &#x12

En el cuadro 2 se detallan las seis entidades alfanuméricas más importantes en *WML:*

Para incluir un carácter especial simplemente utilizaremos la notación descrita anteriormente. Por ejemplo, en el siguiente ejemplo se incluye el caracter mayor que (>) en la cadena de texto:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
    "-//WAPFORUM//DTD WML
    1.1//EN"
"http://www.wapforum.org/DTD
    /wml_1.1.xml">
<wml>
<card id="Card_1">
<p>
Matemáticamente: 3 &#62; 1 y 3
    &#61; 3
</p>
</card>
</wml>
```

En la próxima entrega veremos, entre otras cosas, el funcionamiento de las variables utilizadas para ampliar las prestaciones de los móviles *WAP*.



LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

A SÓLO PROGRAMADORES

QUE NO TE LÍEN CON EL <CÓDIGO>

- NOTICIAS
- MULTIMEDIA
- TEORÍA
- INTERNET
- DESARROLLO DE APLICACIONES
- COMUNICACIONES
- HERRAMIENTAS
- PROGRAMACIÓN
- ÚLTIMAS TENDENCIAS
- Y MUCHO MÁS

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revista SÓLO PROGRAMADORES desde el Número ....
y beneficiarme de las condiciones de esta magnífica promoción:

❏ **SUSCRIPCIÓN ANUAL**
11 NÚMEROS + 11 CD-ROMs
AL PRECIO DE
8.580 ptas. / 51,57€

Precio de suscripción para el extranjero:
11 NÚMEROS + 11 CD-ROMs
AL PRECIO DE
11.800 Ptas.

FORMAS DE PAGO:

❏ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❏ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❏ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❏ Domiciliación bancaria
❏ Contra reembolso

Soy antiguo suscriptor
❏ Sí ❏ No

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:

❏ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❏ Eurocheque conformado con un banco español a nombre de REVISTAS PROFESIONALES, S.L
❏ Por VISA ____/____/____/____
Caducidad: __/__

NOMBRE Y APELLIDOS:..

EDAD:............... PROFESIÓN: ....

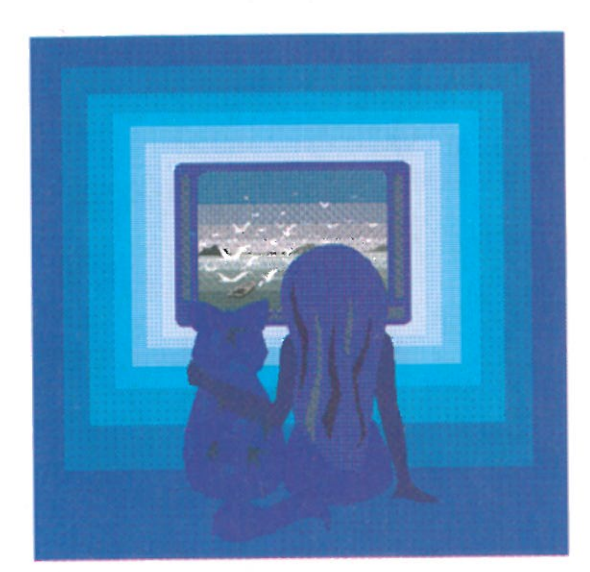
TFNO: .... DOMICILIO: ....

CIUDAD:.............................. C.P: .... PROVINCIA:....................

Datos de domiciliación:
Banco..
Domicilio ....
Nº de Cuenta..................|..................|..................|..................
Titular..
Fecha ....
FIRMA

Promoción válida hasta agotar existencias

Programación OpenGL en Delphi (II)



Juan Luis Ceada Ramos.
Ingeniero Superior de Informática.
Emilio Nestal Díaz.
Ingeniero Técnico en Informática.

El cálculo matricial permite manipular objetos e imágenes en dos dimensiones, de manera que ofrezcan el aspecto de *3D*. Para ello, transformaremos la proyección, ortogonal y perspectiva, y el modelado. Algo que aprenderemos a lo largo de este artículo.

INTRODUCCIÓN

En esta entrega vamos a ver cuáles son las funciones que *OpenGL* pone a nuestra disposición para poder realizar las operaciones más comunes sobre los objetos. Supongamos que estamos realizando un programa de modelado *3D*. En dicho programa, creamos el modelo de un deportivo. Es imprescindible que el usuario del programa pueda mover el coche, rotarlo o escalarlo, las veces que desee, y que esas transformaciones se reflejen automáticamente en la pantalla del ordenador.

Para poder aplicar estas transformaciones sobre los objetos, necesitamos de un mecanismo matemático que nos ayude a realizar estas operaciones. En este caso, como en tantos otros, tendremos que recurrir al cálculo matricial. Gracias a las matrices, las operaciones que se deben realizar son siempre las mismas, tanto para mover un objeto por el espacio, como para rotarlo, por ejemplo.

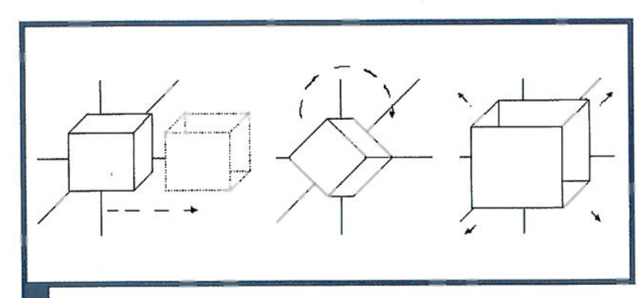


Figura 1.- Diferentes transformaciones.



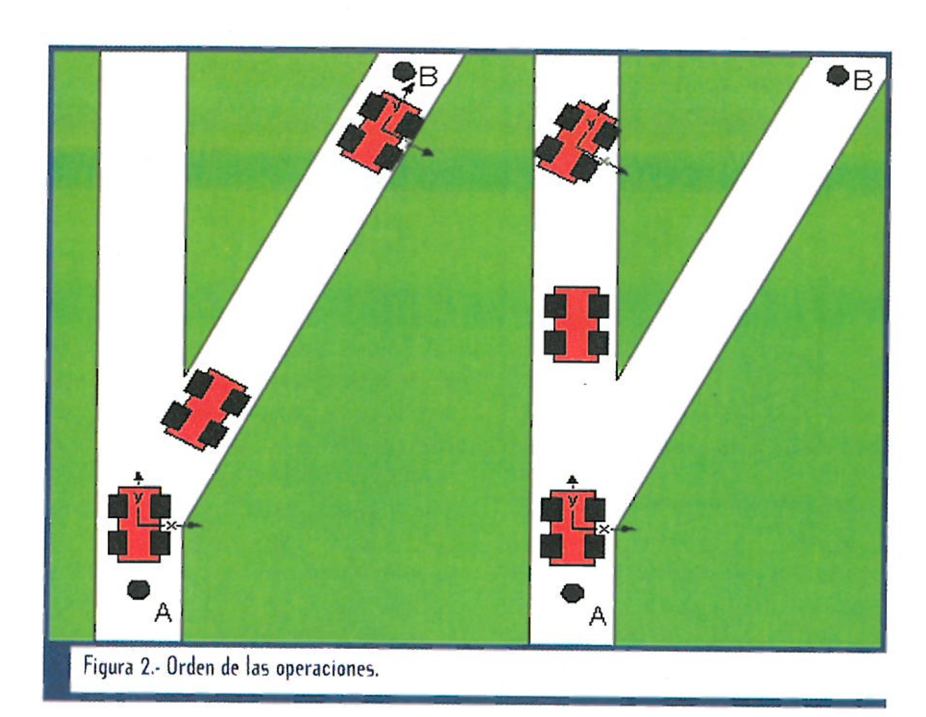


Figura 2.- Orden de las operaciones.

Simplemente, tomaremos las coordenadas de cada uno de los vértices de objeto y las representaremos mediante una matriz, que multiplicaremos por la matriz de transformación. El resultado es una nueva matriz que representa las coordenadas de dicho vértice después de la transformación.

Todas las transformaciones se basan en el cálculo matricial

Dependiendo de cómo esté construida dicha matriz de transformación, los resultados serán unos u otros.

CONOCIMIENTOS MATEMÁTICOS

Para poder seguir esta entrega, no son necesarios grandes conocimientos matemáticos (aunque nunca están de más). Basta con saber qué es una matriz y parte de la terminología asociada a ellas (como por ejemplo, qué es una matriz identidad).

No vamos a tener que realizar nuestros propios algoritmos de multiplicación de matrices, puesto que *OpenGL* proporciona funciones que hacen estas operaciones por nosotros. Pero saber cómo se trabaja con matrices, nos puede ayudar a entender con más facilidad los conceptos que vamos a exponer en esta entrega de la serie.

TRANSFORMACIONES

Las transformaciones realizadas mediante el cálculo matricial hacen posible que podamos representar escenas *3D* en un medio con sólo *2D*, como es la pantalla del ordenador. Pero, además, las transformaciones también permiten rotar los objetos, moverlos, estirarlos y encogerlos.

Son dos las principales transformaciones que vamos a poder hacer: de modelado y de proyección.

TRANSFORMACIÓN DEL MODELADO

Se usa para manipular los objetos de nuestra escena. Coloca en su lugar los objetos, los gira y los escala. En la Figura 1 se muestran los resultados obtenidos después de aplicar algunas transformaciones sobre un objeto.

Empezando por la izquierda, tenemos el objeto trasladado una determinada distancia en el eje *X*. En segundo lugar, el objeto se ha rotado "x" grados. Y por último, hemos escalado el objeto.

La operación de escalado puede realizarse de forma individual en cada eje, con lo que podemos aumentar o disminuir el tamaño de un objeto (aplicando la misma escala a todos los ejes), o "engordarlo" y "adelgazarlo" (cambiando de escala únicamente uno de los ejes).

Es importante indicar que el orden de las operaciones influye en el resultado final de la escena. En la Figura 2 tenemos un ejemplo. El coche debe ir desde el punto *A* al punto *B*, siguiendo la carretera. En todo momento, el eje *Y* viene indicado por el capó del vehículo. Si primero trasladamos en el eje *Y*, y después giramos, no llegaremos al destino. Es necesario girar primero el sistema de coordenadas, y trasladar después.

LA TRANSFORMACIÓN DE LA PROYECCIÓN

Como vimos en la pasada entrega, existen dos tipos de proyección: ortogonal y en perspectiva. Este tipo de transformación define cómo se traslada una escena ya modelada a la pantalla.

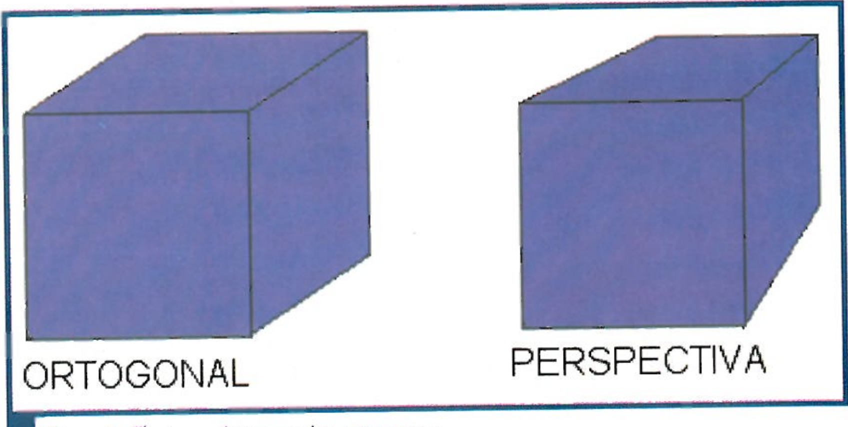


Figura 3.- El mismo objeto con dos perspectivas.

En la proyección ortogonal, todos los polígonos se dibujan en pantalla exactamente con las dimensiones relativas especificadas (sin tener en cuenta que polígonos más lejanos se verían más pequeños). En la proyección en perspectiva, ajustar el tamaño de los objetos en función a su lejanía, da una sensación más real, puesto que es así como vemos las cosas en la vida real. Recurramos a un ejemplo típico para mostrar este efecto: las vías del tren.

La proyección en perspectiva ajusta el tamaño de los objetos según su lejanía

Todo el mundo sabe que las vías del tren son paralelas, desde su comienzo hasta su fin. Pero si nos colocamos entre dos vías, y miramos a lo lejos, nos da la impresión de que las vías se unen en un punto, situado más o menos donde alcanza nuestra vista. A ese punto se le conoce como punto de desvanecimiento.

Supongamos que queremos dibujar un cubo. Hacemos dos cuadrados, y a uno de ellos lo desplazamos un poco hacia la izquierda, y lo alejamos. Después unimos los vértices, y ya tenemos nuestro cubo. Ahora le indicamos a *OpenGL* que lo muestre en proyección ortogonal: la cara frontal y trasera del cubo son dos cuadrados, del mismo tamaño, que dan la sensación de que forman un cubo (después de unir algunos vértices).

Ahora le indicamos a *OpenGl* que lo muestre en perspectiva. Automáticamente, a todos los vértices se les aplica una transformación (internamente son multiplicaciones de matrices), de forma que el cuadrado de la parte de atrás es más pequeño que el de la parte delantera, obteniendo de esta forma, la sensación de lejanía (ver Figura 3).

¿Cómo se efectúan las transformaciones?

La matriz de modelado (en adelante, *MM*) es una matriz de *4 x 4*, cuyos valores determinan el tipo de transformación que aplicar. Las coordenadas de los vértices se introducen en una matriz de una columna y cuatro filas (una fila por coordenada, más un cuarto valor, se utiliza para normalizar, y no suele modificarse), que se multiplica por la *MM*. Esto produce como resultado una nueva matriz de *4 x 1*, que contiene las nuevas coordenadas del vértice una vez transformado.

Por supuesto, todo este proceso es automático, y no nos tendremos que preocupar por introducir los vértices en las matrices, multiplicarlas, etc. *OpenGL* lo hace por nosotros.

Como hemos dicho, los valores de *MM* definen la transformación que aplicar. En la Figura 4 se mues-

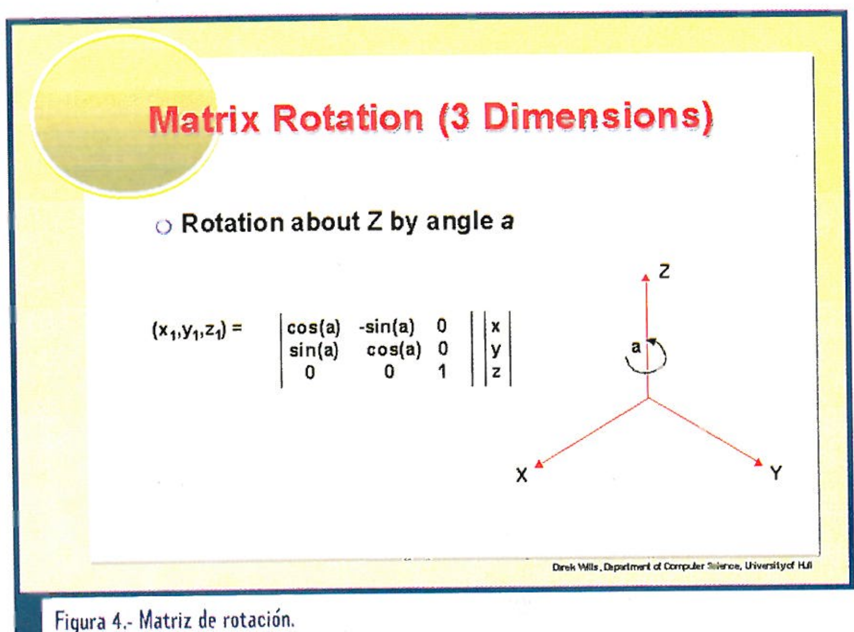


Figura 4.- Matriz de rotación.



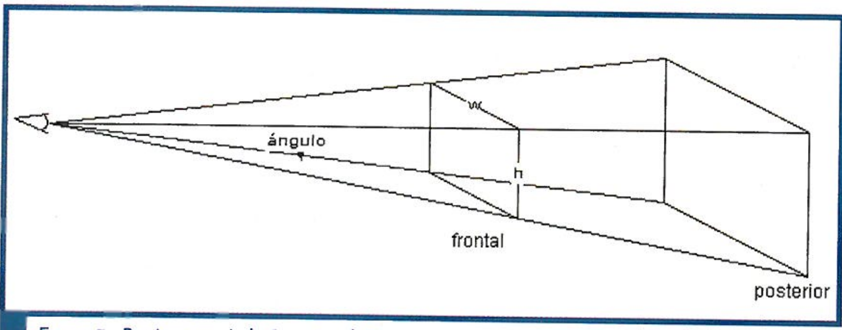


Figura 5.- Parámetros de la función *gluPerspective.*

tra la matriz necesaria para rotar el ángulo *Z* un valor de "a" grados.

INICIAR LA MATRIZ

¿Cómo hacemos para que *MM* haga una transformación u otra?

Mediante una serie de funciones que veremos a continuación:

- *glTranslatef(x, y , z: GLfloat).* Construye una matriz de traslación que se multiplica por la matriz *MM*. Los valores de desplazamiento en cada uno de los ejes vienen indicados por los parámetros *x* y *z*.
- *GlRotatef(ángulo, x, y, z:GLfloat).* Construye una matriz de rotación que se multiplica por *MM*. Los parámetros *x, y, z* indican un vector de dirección. Dicho vector es una línea recta que va desde el origen al punto de coordenadas *x, y, z*.
- *glScalef(escalax, escalay, escalaz:GLfloat).* Multiplica la *MM* por una matriz de escalado.

Inicialmente, *MM* equivale a la matriz unidad. Al multiplicar las coordenadas de un vértice por la matriz unidad, obtenemos el mismo vértice. Cuando ejecutamos algunas de las funciones anteriores, se multiplica *MM* por una nueva matriz diseñada para efectuar la transformación deseada, y el resultado se guarda en la matriz *MM*, que permanece activa hasta que la cambiemos.

El siguiente ejemplo realizaría la rotación y posterior traslación de un cubo:

```
glRotatef(angulo, 0.0, 1.0, 0.0);
glTranslate(0.0, 10.0, 0.0);
DibujarCubo;
```

Las operaciones con las matrices del modelador son acumulativas

Hay que tener en cuenta un pequeño detalle: las dos instrucciones *GL* han modificado de tal forma la matriz *MM,* que a partir de ahora, cualquier objeto que dibujemos, aparecerá rotado y trasladado. En efecto, las transformaciones son acumulativas. El siguiente pseudocódigo muestra el estado actual de la matriz.

```
MM = Matriz_Unidad
MM = MM * Matriz_Rotacion
     (angulo,0,1,0)
MM = MM * Matriz_Traslacion
     (0,10,0)
```

¿Y qué pasa si queremos dibujar un cubo rotado, y otro normal? El siguiente código es incorrecto, ya que dibujaría los dos cubos rotados.

```
glRotatef(angulo, 0.0, 1.0, 0.0);
DibujarCubo;
DibujarCubo2;
```

De alguna forma tenemos que reiniciar *MM*, para que vuelva a equivaler a la matriz unidad, de forma que no se aplique ninguna transformación sobre el objeto. Para ello, tendremos que recurrir a la función *glLoadIndenty().* El código correcto es el siguiente:

```
glRotatef(angulo, 0.0, 1.0, 0.0);
DibujarCubo;
GlLoadIndetity;
//este cubo no es rotado
DibujarCubo2;
```

PILAS DE MATRICES

A veces nos puede interesar almacenar de alguna forma el valor actual de *MM* antes de iniciarla, mediante una llamada a *GlLoadIndetity.* Para ello *OpenGL* pone a nuestra disposición las funciones *glPushMatrix*() y *glPopMatrix*(), que sirven para almacenar y extraer de una pila los valores de *MM*. Supongamos el siguiente código:

```
glRotatef(angulo, 0.0, 1.0, 0.0);
DibujarCubo;
GlPushMatrix;
GlLoadIndetity;
DibujarCubo2;
GlPopMatrix;
DibujarTriangulo;
```

En este caso, el primer cubo se muestra rotado con un determinado ángulo. El segundo cubo se muestra "normal". El triángulo se dibujará con la misma rotación que

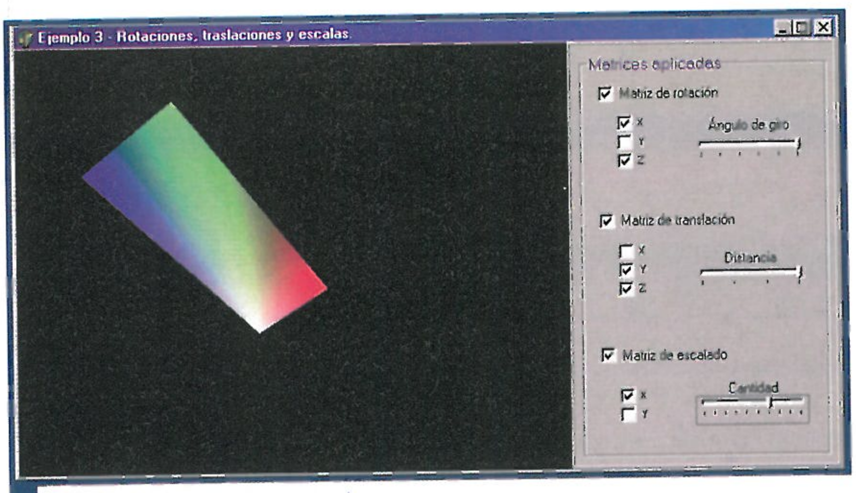


Figura 6.- Aspecto del primer ejemplo

el primer cubo, puesto que antes de iniciar *MM*, tomamos la precaución de almacenarla en la pila, y la hemos restaurado antes de dibujar el triángulo.

PROYECCIONES

Cambiar la proyección de nuestra escena es tan sencillo como llamar a un par de funciones: *glOrtho(xmin, xmax, ymin, ymax, zmin, zmax:GLFloat)* y *gluPerspective(angulo, aspecto, cerca, lejos:GlDouble)*. La primera de ellas hace que la escena se dibuje usando proyección ortogonal. Como parámetros, recibe las coordenadas del área de visualización y la profundidad. La llamada más típica a esta función tiene el siguiente formato:

```
glOrtho(0, ClientWidth,0,
    ClientHeight, 1.0, -1.0).
```

La segunda hace que la escena se dibuje usando proyección en perspectiva. Los parámetros indican el ángulo de visión vertical, la relación de aspecto entre altura y anchura, y la distancia entre los planos posterior y frontal. En la Figura 5 se muestra a qué equivalen estos parámetros.

Los pasos a seguir para cambiar de perspectiva se muestran en el siguiente código:

```
GlMatrixMode(GL_PROJECTION);
GlLoadIdentity;
GluPerspective(60,10,1,400)
GlMatrixMode(GL_MODELVIEW);
GlRotate(10,10,0);
DibujarCubo;
```

En primer lugar, hay que indicar de alguna forma, que ahora trabajamos con la matriz de proyección, y no con la de modelado. Para ello, usamos la función *glMatrixMode*, con el parámetro *GL_PROJECTION*. Después, iniciamos la matriz de proyección. Si no llamamos a *glLoadIdentity*, puede que la perspectiva que obtengamos no sea la que esperábamos, ya que, al igual que las transformaciones, los cambios de perspectiva son acumulativos.

Podemos almacenar el estado de una matriz MM en una pila

Por último, volvemos a trabajar con la matriz de modelado, realizamos una rotación, y dibujamos un cubo.

EN LA PRÁCTICA

En los ejemplos que vamos a comentar a continuación, veremos cómo utilizar las funciones de alto nivel para la realización de rotaciones, traslaciones y escalados, y cómo trabajar con las pilas de matrices.

ROTACIONES, TRASLACIONES Y ESCALADOS

Tal como se ha visto en las secciones anteriores, *OpenGL* suministra funciones de alto nivel que nos

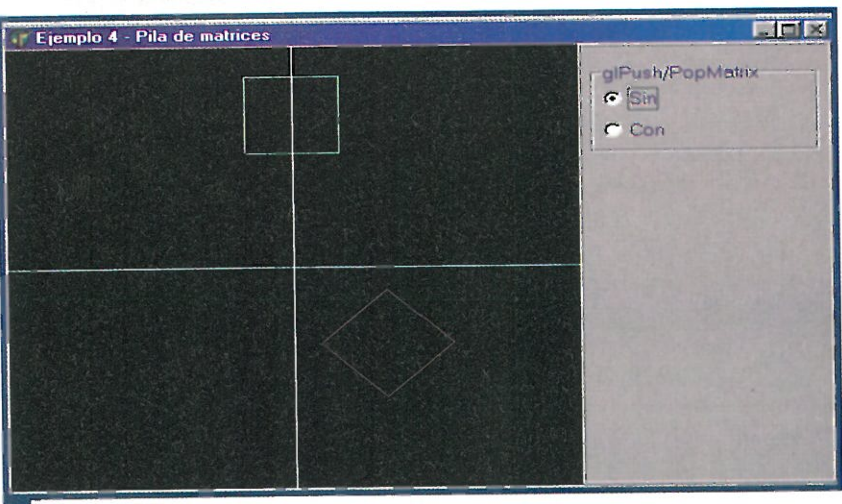
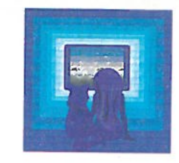


Figura 7.- Ejemplo 1: Trabajo sin *Push/PopMatrix*.



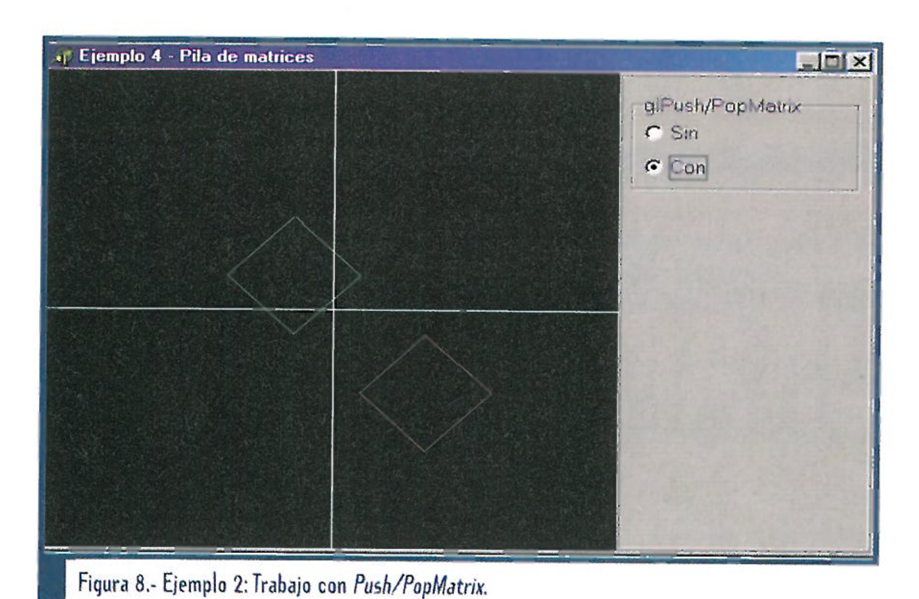


Figura 8.- Ejemplo 2: Trabajo con *Push/PopMatrix*.

permiten obviar los detalles matemáticos de las operaciones matriciales necesarias para rotar, trasladar y escalar objetos.

La API OpenGL permite realizar operaciones en 3D obviando los conceptos matemáticos

El primer ejemplo, permite ver cómo podemos utilizar dichas funciones. Se ha incluido la posibilidad de rotar, trasladar y escalar el cuadrado que vemos en pantalla, pero esta vez lo que se mueve es el espacio, y no la cámara como se vio en la entrega anterior.

Estas operaciones se realizan con unas sencillas funciones *OpenGL*. Los controles presentes en el formulario, permiten cambiar los parámetros de las funciones de transformación en tiempo de ejecución.

Concretamente hemos utilizado *glRotate(Angulo, ejeX, ejeY, ejeZ) glTranslate(DisX,DisY,DisZ)* y *glScale(AspecX, AspecY, AspecZ)*. Como podréis comprobar, son muy sencillas de utilizar.

Como hemos comentado, el orden en el que se ejecuten estas operaciones influye en el resultado final, es decir, no obtenemos el mismo resultado mediante un desplazamiento y una rotación, que con una rotación y un posterior desplazamiento, por lo que es conveniente establecer un orden. En estos ejemplos y en los posteriores usaremos el orden rotación – desplazamiento.

PILAS DE MATRICES: USO PRÁCTICO

En el segundo ejemplo vamos a ver cómo podemos trabajar con la pila de matrices y cómo influirá en el resultado de nuestras representaciones.

En este caso vamos a ver cómo podemos utilizar la pila de matrices del modelador, para colocar dos objetos. El primero de ellos, un cuadrado rojo, al cual se le aplicarán una serie de operaciones, y otro un cuadrado verde, que en función de cómo trabajemos con la pila de matrices se verá o no afectado por dichas transformaciones.

Para ello, lo primero que hacemos es almacenar la matriz del modelador, mediante la función *glPushMatrix*, que almacenará la matriz activa en la cima de la pila de matrices del modelador, en este momento, la matriz identidad. A partir de aquí, modificaremos dicha matriz, mediante un *glTranslate* y un *glRotate*, para más tarde, dibujar el primer objeto, el cuadrado rojo. A continuación, tenemos dos opciones, restablecer la pila mediante un *glPopMatrix* o no hacerlo.

La posibilidad de restablecer la matriz almacenada va a condicionar la posición del segundo objeto, es decir, si no la restablecemos, el resultado de las operaciones sobre el cuadro rojo afectarán también al verde. En caso contrario, si restablecemos la matriz, las dos operaciones que se realizaron sobre el rojo no afectan al verde.

El uso de la pila de matrices nos permitirá trabajar cómodamente sin necesidad de almacenar matrices de operaciones de forma temporal ni tener que deshacer operaciones para dejar la matriz del modelador conforme a nuestras necesidades.

EN LA PRÓXIMA ENTREGA...

Iniciaremos el trabajo con listas de visualización, que nos permitirán el aumento de prestaciones temporales, gracias a la organización de las operaciones gráficas por parte del controlador *OpenGL*, con lo que podremos llegar a mejorar nuestras aplicaciones hasta en un 200% desde el punto de vista de la eficiencia temporal.

Además, comenzaremos con el modelado avanzado, concretamente con las cuádricas que suministra nativamente *OpenGL*.

¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

ATENCIÓN

Los números 49 al 51, ambos inclusive, se encuentran AGOTADOS







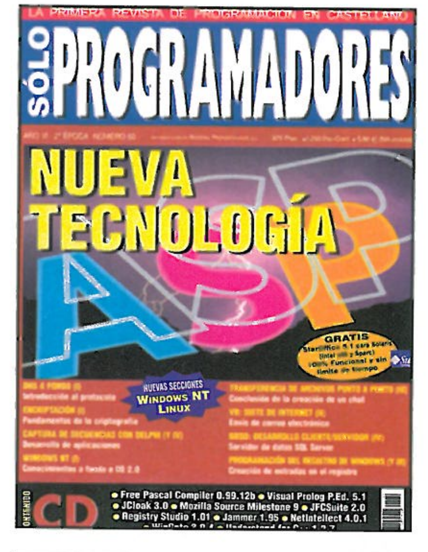






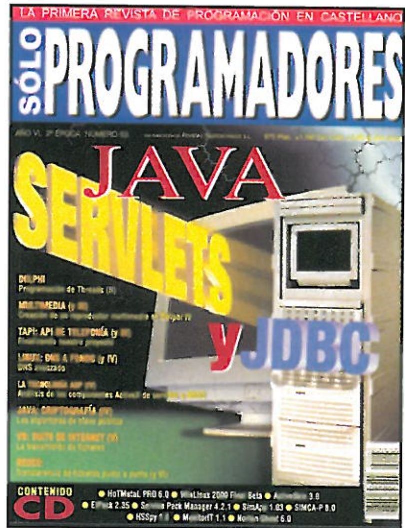












BOLETÍN DE PEDIDOS

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me manden los número/s..................................
....
....

NOMBRE Y APELLIDOS:...
....
EDAD:........... PROFESIÓN: ....TFNO: ....
DOMICILIO: ....
CIUDAD: ....
PROVINCIA: ....C.P: ....

❑ **Precio por unidad: 975 pesetas** + 700 ptas. de gastos de envío.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

DATOS DE DOMICILIACIÓN:

Banco...

FIRMA

Domicilio...
Nº de Cuenta...............|...............|...............|...............
Titular...
Fecha..



VisualCafé 4 Vs. JBuilder 3

Javier Sanz Alamillo.
Ingeniero de Software.

En el número anterior os presentamos *VisualCafé 4*. Ha llegado la hora de compararlo con uno de los entornos de desarrollo en *Java* más asentados: *JBuilder 3*. Esperamos que esta comparativa sirva de ayuda a los desarrolladores para elegir entre uno u otro.

JBUILDER 3

La respuesta de *Borland* –ahora en el grupo *Inprise*, adquirido por *Corel*- al mercado *Java* es *JBuilder 3*, un completo entorno de desarrollo que hará las delicias de los seguidores de los productos *Borland*.

JBuilder se nos presenta como el primer entorno basado en *Java 2*. Integra todo lo necesario para realizar la mayoría de los desarrollos actuales, y supone una importante mejora en todo lo relacionado con las bases de datos y el entorno visual, además de ofrecer una gran integración con la programación distribuida (*RMI/CORBA*).

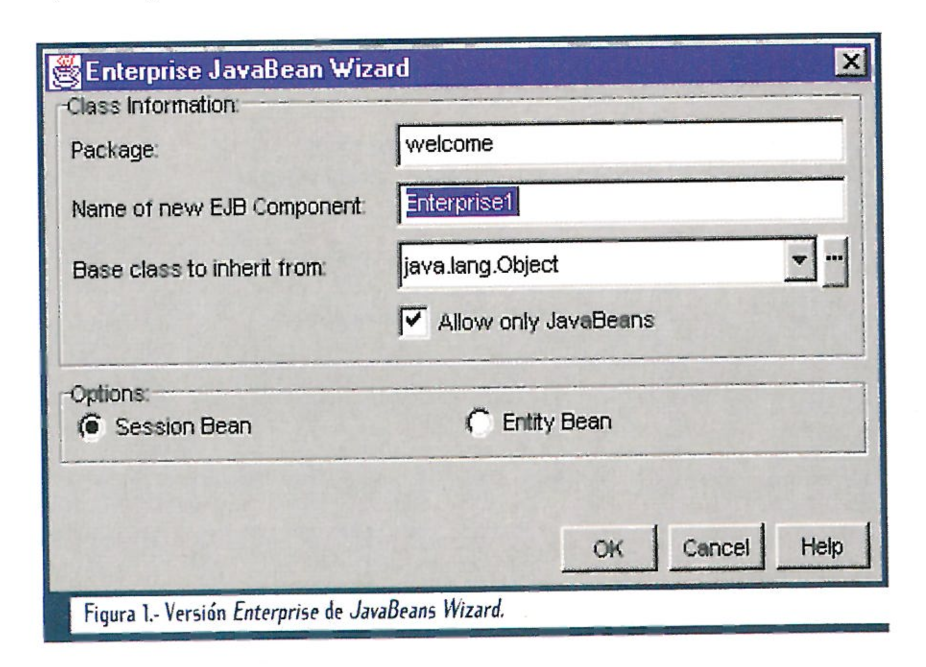


Figura 1.- Versión *Enterprise* de *JavaBeans Wizard*.

INSTALACIÓN

Al igual que pasaba en la instalación de *VisualCafé 4,* el proceso de instalación de *JBuilder* 3 es sumamente sencillo, y disponiendo del suficiente tamaño en disco basta con seguir las indicaciones del asistente, finalizando en escasos minutos.

REQUERIMIENTOS

JBuilder 3 actualmente está disponible para *Windows 95, 98 y NT*, y



para alegría de los programadores de entornos *Solaris* y *Linux*, se está finalizando la edición *Enterprise* para esos entornos.

Su presentación ha hecho que *JBuilder* gane un importante sector de mercado, lo que provocará que el entorno se mejore pasado un cierto tiempo por los consejos y recomendaciones de los expertos.

Básicamente, los requerimientos *hardware* son los siguientes:

- *Pentium* a 133 *Mhz* o superior para su ejecución. Al igual que con *VisualCafé*, el uso de algo menor a un *Pentium II* a 300 *Mhz* queda algo justo, por lo que no es muy recomendable nada menor a 450 *Mhz*.
- 128 *Mb* de memoria *RAM*. Se debería seguir la recomendación.
- Como tarjeta de vídeo se recomienda una *SVGA*.

CARACTERÍSTICAS GENERALES

Hasta hoy, la familia de entornos *JBuilder* no ha tenido una gran aceptación por parte de los desarrolladores, teniendo en cuenta que la competencia ofrece productos muy serios, principalmente *IBM* con su abanderado *VisualAge* y *Symantec* con *VisualCafé*. A partir de esta versión 3, más de un programador se planteará de verdad basar el desarrollo de sus aplicaciones en *JBuilder 3*.

Si todas las referencias a anteriores versiones presentaban un producto algo inmaduro, con un aspecto visual incómodo e incompleto, en este caso se ha producido un vuelco total.

Son notables las continuas mejoras que *Borland* ha ido aplicando a *JBuilder* a partir de las sugerencias de usuarios y programadores. El resultado ha sido un entorno que permite tanto a desarrolladores individuales como a empresas de *software* crear aplicaciones *Java* potentes usando esta herramienta.

De hecho, fueron los primeros en ofrecer un producto que se basara en *Java 2*, con lo que parte de ese mercado que estaba esperando una herramienta ha sido "adoptado" por *Borland*.

El entorno de desarrollo de JBuilder ofrece una apariencia más profesional

El conjunto de cambios y mejoras en la herramienta es sustancioso, donde destacan los cambios producidos en el conjunto de los diversos asistentes por encima de la interfaz. De esta manera el usuario obtiene una mayor comodidad, facilidad y rapidez en el trabajo.

Se ha mejorado el tiempo de ejecución y compilación, permitiéndose el uso de diferentes *JDK*, así como el proceso de depuración.

Se ofrece una librería con más de 300 *JavaBeans*, y el asistente para su creación ha sido mejorado y representado de manera más racional, aunque ya de por sí era uno de los más completos para la creación de *beans*.

Mediante el desarrollo visual se pueden construir *applets*, aplicaciones, *beans*, *EJB* y aplicaciones *CORBA*, con lo que al igual que ocurría con *VisualCafé* se sigue la política de alejar al programador del código evitando así posibles equivocaciones o descuidos.

La posibilidad de crear aplicaciones con bases de datos se ha ampliado en cuanto al número de asistentes, añadiéndose herramientas para la creación visual de bases de datos, de modelado de datos y más *drivers JDBC*, etc.

La integración para realizar aplicaciones *CORBA* está asegurada por su relación con *Visigenic*, también del grupo *Inprise*, por lo que realizar aplicaciones *CORBA* se convierte en algo relativamente sencillo.

Se dispone de asistentes y herramientas para realizar la ma-

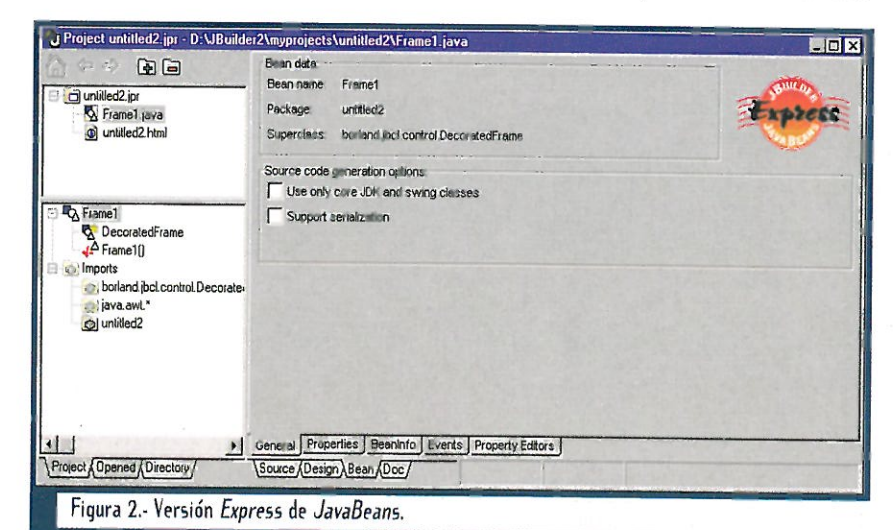


Figura 2.- Versión *Express* de *JavaBeans*.

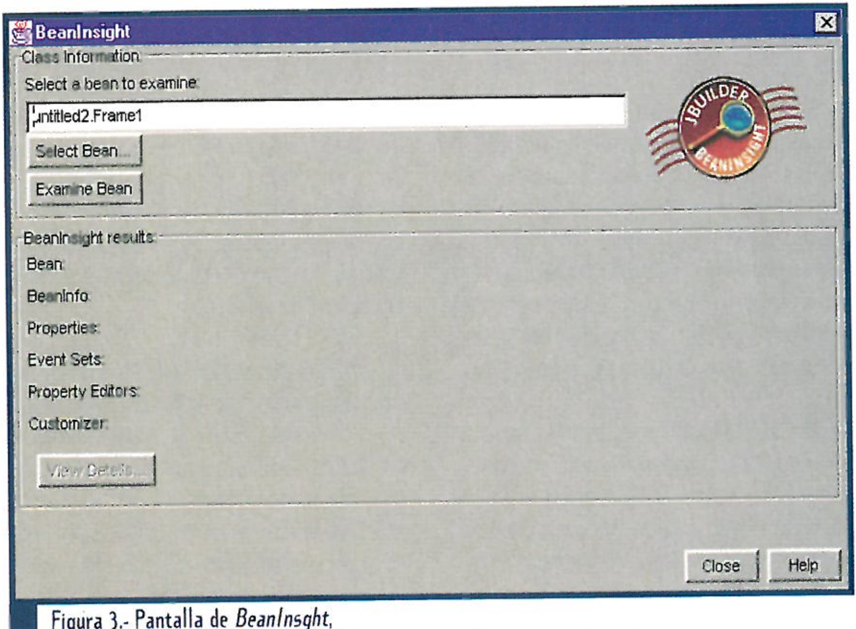


Figura 3.- Pantalla de *BeanInsght*,

yoría de las tareas (*Application Generator Wizard, Data Modeler Tool*, etc.) como es la creación de *IDL* visualmente y la automatización para la creación de determinados tipos de aplicaciones, etc. Además, también se puede utilizar fácilmente *OrbixWeb* de *IONA*, por lo que no hay excusa a la hora de desechar este producto porque esté basado en un *ORB* no esperado.

Se ha mejorado en relación al uso de *JFC/Swing*, no sólo en lo referente al número de componentes, sino a su velocidad y facilidad de uso. Ahora las propiedades y eventos se pueden gestionar con más facilidad.

La edición Profesional es sumamente completa

El nuevo *JIT* (*Just in Time Compiler*) hace que el código se ejecute más rápidamente, estando a la altura de los mejores del mercado.

TABLA 1. Comparación entre las tres ediciones

Característica	Estándar	Profesional	Enterprise
Mejoras con *Java 2*	X	X	X
JFC/Swing Java2	X	X	X
Nuevos asistentes	X	X	X
Java Help, easy Deploy	X	X	X
Gestor de proyecto	X	X	X
JIT, compilador mejorado	X	X	X
Asistente *JavaDoc*		X	X
Internacionalización		X	X
Mejoras en *JDBC* y *BBDD*		X	X
Asistentes *JavaBeans*		X	X
Control de versiones			X
Soporte *CORBA*			X
Creación de *EJB*			X

Con la inclusión del *SmartChecker*, se compila ahora mucho más rápido y de forma incremental.

Para completar el conjunto de cambios y mejoras, se incluyen herramientas como un *obfuscator* (impide en cierto modo la descompilacion de las clases), utilidades estilo *UNIX* como *grep, touch, make*, etc., que hacen que el paquete sea más completo de cara al programador que si sólo obtuviéramos un entorno de desarrollo.

EDICIONES DISPONIBLES CON JBUILDER 3

J*Builder 3* se presenta en tres diferentes ediciones: Estándar, Profesional y *Enterprise*, que se adaptan a las necesidades del proyecto que se va a realizar.

EDICIÓN ESTÁNDAR

Con esta versión nos adentramos en un entorno básico y sencillo de utilizar, permitiendo la creación de *applets*, aplicaciones, *beans* en *Java 2* de forma muy rápida. Se dispone de un entorno visual potente y eficaz, apoyado por características de *drag-and-drop* y de asistentes que facilitan la realización de muchas tareas, como son *CodeInsight* y *BeanInsight*.

La creación de *JAR* se ha mejorado, corrigiendo algunas anomalías en el tratamiento de las *inner classes*, serialización, etc.

Las aplicaciones visuales son ahora *Java Puro 100%*, y no utiliza código propietario de *Borland* en, por ejemplo, los *beans* que se puedan generar. Al utilizar *Java 2* se



pueden acceder a nuevas posibilidades: *Graphics 2D, Collections*, etc.

El acceso a la ayuda se ha mejorado y ampliado, con multiplicidad de asistentes y con un funcionamiento más rápido.

EDICIÓN PROFESIONAL

Esta edición está indicada para los desarrolladores de aplicaciones con un determinado volumen, que requieren de un completo entorno para la gestión de bases de datos, distribución, etc.

Están presentes las características de la anterior edición. Pero añade la posibilidad de crear *Servlets*. Se dispone además de una biblioteca de más de 300 *beans* y un singular *drag-and-drop* para realizar aplicaciones *JFC /Swing/dbSwing*, que permite la integración de resultados provenientes de bases de datos directamente en componentes, todo ello de una forma automática y que se puede personalizar con cierta facilidad.

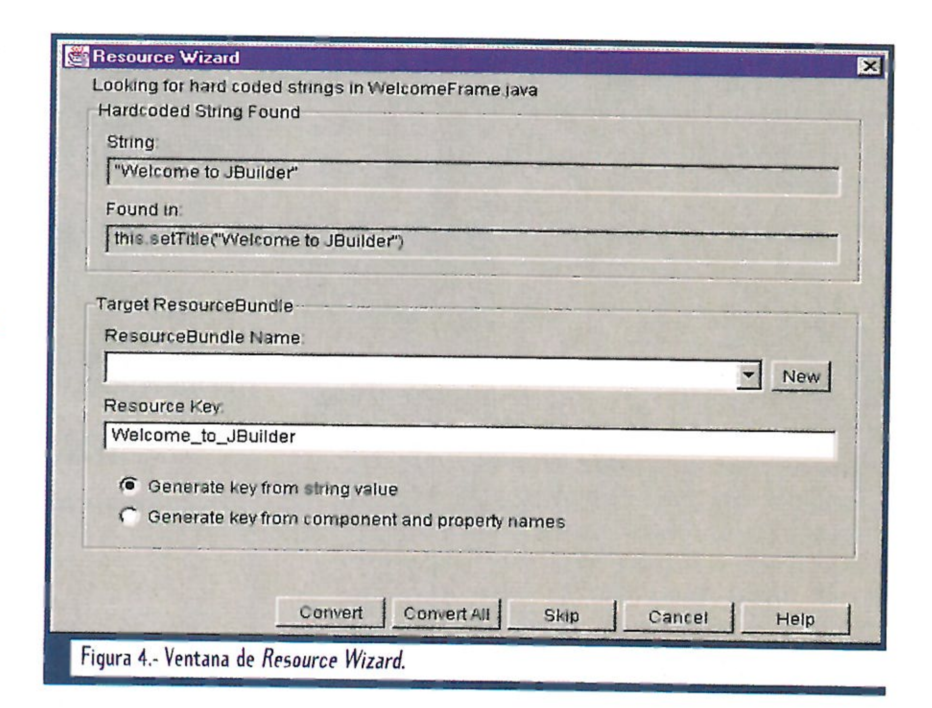


Figura 4.- Ventana de *Resource Wizard*.

Ambos entornos de desarrollo resultan muy completos

En esta edición encontramos un conjunto de herramientas añadidas, entre las que destaca la posibilidad de realizar diseños de bases de datos de manera visual.

Cuenta además con un asistente que permite la migración de aplicaciones de *Java 1.1* a *Java 2*, y un visor de *HTML* integrado en el entorno.

EDICIÓN ENTERPRISE

De las tres ediciones, ésta es la más completa de todas, por tanto ideal para aquellos que vayan a realizar aplicaciones de ámbito empresarial. Contiene todas las características de las anteriores ediciones añadiendo las siguientes novedades:

- Consta de todo un conjunto de posibilidades para desarrollar aplicaciones distribuidas, mediante *RMI* y *CORBA* (usando *VisiBroker CORBA ORB*). La creación de *IDL's* se realiza gráficamente mediante la herramienta *DataModeler*.
- El *debugger* de esta edición permite realizar una depuración distribuida de una forma bastante más rápida que en anteriores versiones de *JBuilder*.
- Permite la posibilidad de integrar control de versiones mediante el producto *PVCS*.
- Contiene un amplio conjunto de *drivers JDBC* y características para crear aplicaciones con bases de datos que cumplan con el estándar *SQL-92*.
- Mediante *Pure Java Jdata Store* se permite gestionar la caché de datos y persistencia de datos, objetos y ficheros planos.
- Permite la creación de *EJB* (*Enterprise Java Beans*) de forma más o menos sencilla.

Todas las posibilidades antes mencionadas ofrecen un entorno muy completo y preparado para realizar cualquier tipo de aplicación actual, desde *e-commerce* hasta cierto tipo de aplicaciones en tiempo real.

La Tabla 1 muestra las principales características que ofrecen las tres ediciones. Así, podremos elegir cuál es la edición que mejor se adapta a nuestras necesidades.

Para finalizar, recordar que se puede adquirir *JBuilder* 3 edición Estándar por 99$, la edición Profesional por 799$ y la versión *Enterprise* por 2499$.

Anunciamos en este artículo, que está prevista la liberación de

una edición *Enterprise* para entornos *Solaris* y *Linux*, lo que hará que otros fabricantes se animen a adaptar sus entornos a estas plataformas.

VISUALCAFÉ Vs. JBUILDER 3

Como se puede comprobar, los dos entornos son muy completos y ofrecen prestaciones similares. Con respecto a las anteriores versiones, observamos que se han ampliado y modificado muchas facetas, lo que les ha hecho muy competitivos frente a otros entornos de desarrollo.

Al programador le interesa en gran medida, el hecho de poder elegir entre varios entornos preparados para desarrollar los principales tipos de aplicaciones con gran potencia y calidad.

VisualCafé y *JBuilder* son entornos diseñados para que los programadores *Java* se encuentren cómodos, disponiendo de gran cantidad de ayuda para realizar la mayoría de las tareas. Hay que añadir que, el hecho de prescindir de otras herramientas para realizar su trabajo, y que todo lo que realiza en el entorno, es lo suficientemente válido como para no tener que realizar pruebas incómodas, son innecesarias.

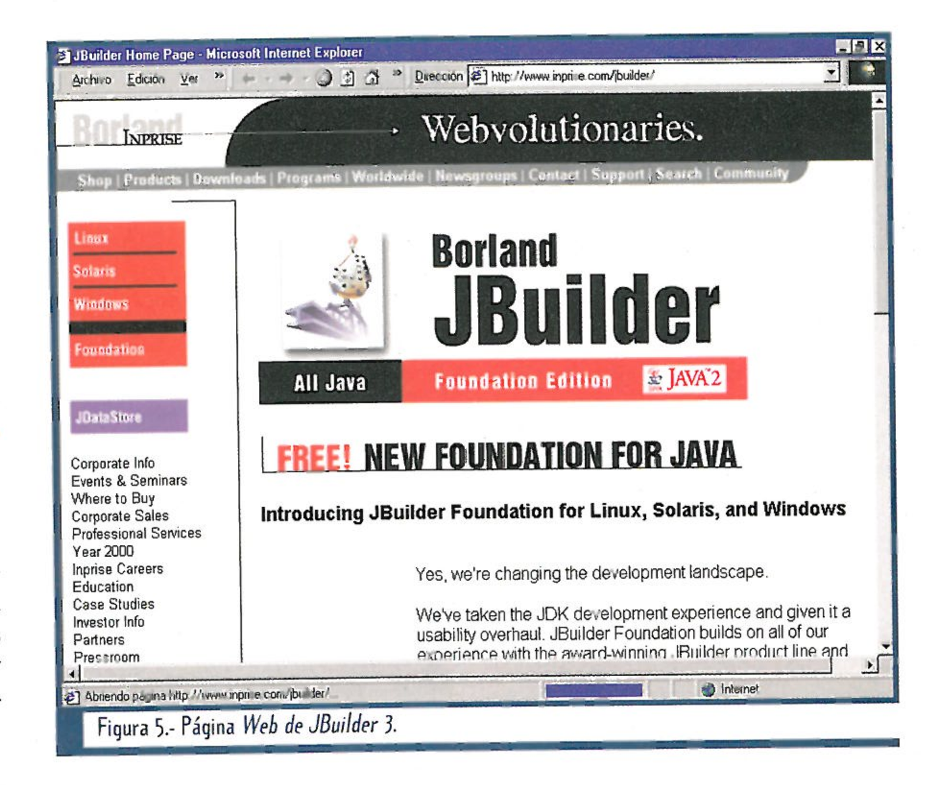


Figura 5.- Página *Web de JBuilder 3.*

Las diferencias entre los dos entornos son mínimas

Ambos entornos ofrecen nuevas posibilidades en relación con *Java 2*, como es el uso de *Collections*, *Graphics 2D*, la última versión de *JFC/Swing 1.1*, así como a la configuración basada en diferentes máquinas virtuales, etc.

Tanto *VisualCafé* como *JBuilder* ofrecen un conversor de aplicaciones *Swing 1.0.3* a *1.1* y garantizan una total libertad en las aplicaciones del uso de clases propias del entorno.

A nivel de entorno, compilador, *debugger* y *JIT*, ambos son simila-

TABLA 2. Resumen de las principales características de ambos entornos

Característica	VisualCafé 4	JBuilder 3
Java 2, Swing, ...	Sí, completo, 100%	Sí, completo, 100%
Velocidad, entorno	El más rápido	Muy rápido
Plataformas	*Windows*	*Windows, Solaris y Linux*
Asistentes múltiples	Sí, para la mayoría de las tareas	Sí, para la mayoría de las tareas
Bases de datos	Completo	Completo
Swing, JavaBeans	Asistentes, código disponible	Múltiples asistentes, código
Distribución *(RMI, CORBA, EJB)*	Asistentes, *debugger*	Asistentes, *debugger*
CORBA	Sí, con *IONA*	Sí, con *Visigenic*
Control versiones	Sí, varios productos	Sí, único producto.
Herramientas apoyo	De terceros, fuera del entorno	Se disponen con el entorno

res, aunque destaca *VisualCafé* por su increíble velocidad de compilación y ejecución. El *debugger* es muy completo en ambos entornos.

La plataforma en la que están disponibles actualmente es *Windows*, pero en unos meses *JBuilder* presentará la edición *Enterprise* en *Solaris* y *Linux*, lo que supondrá un paso adelante para *Borland*.

Borland está preparando ediciones para Solaris y Linux

Todas las tareas relativas a bases de datos están en los dos entornos muy bien resueltas, con los suficientes asistentes y ayudas, y aunque *VisualCafé* con la presentación de *Oracle8i* parece tomar algo de delantera, *JBuilder* lo compensa con más funcionalidades en todo lo relacionado con la creación y gestión de las bases de datos, los *drivers*, etc.

Respecto a la creación de aplicaciones distribuidas, con *RMI* o *CORBA*, ambos entornos lo tienen muy bien resuelto. *VisualCafé* se alía con *IONA* y su *Orbixweb*, el *ORB* en *Java* más maduro del mercado. Por otro lado, *JBuilder* se une con *Visigenic*, que es más "didáctico" a la hora de programarse. Ambos ofrecen facilidades para crear *IDL's*, servidores, etc.

La creación de *Servlets*, *JPS* o *EJB* también se resuelve en ambos entornos utilizando asistentes, aunque en este caso parece que *JBuilder* es algo más completo que *VisualCafé*.

Respecto al control de versiones, *VisualCafé* está más preparado, ya que permite la utilización de diferentes productos. Y en este caso es algo superior a *JBuilder*.

VisualCafé está más preparado en el control de versiones

Por otro lado, *JBuilder* ofrece fuera del entorno de desarrollo un conjunto de herramientas que lo complementan, como un *obfuscator*, un *shrinker*, etc., lo que hace que para muchas tareas el programador se sienta más protegido. No se tiene la sensación de que sólo tiene un entorno, se puede decir que es el valor añadido de *JBuilder*.

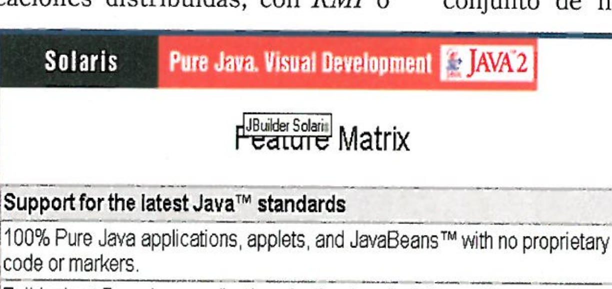


Figura 6.- *JBuilder para Solaris.*

Como se puede apreciar, los dos entornos son sumamente completos y ofrecen características similares, incluso hasta en el precio. Los desarrolladores que hayan utilizado anteriores versiones de *VisualCafé* no dudarán en continuar usando *VisualCafé 4* y los que hayan comenzado usando *JBuilder* se decidirán por la versión 3.

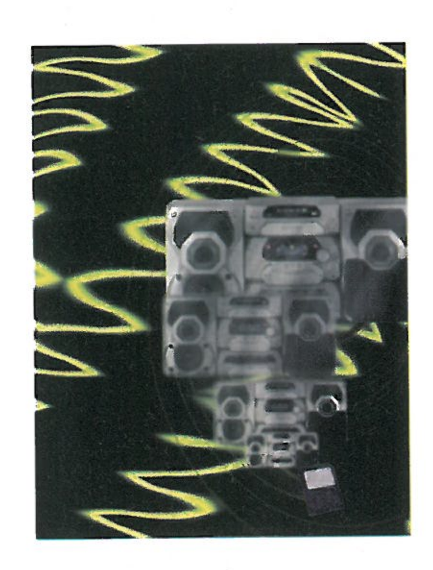
Casi se puede llegar a decir que la elección puede basarse en la familiaridad que se haya tenido con productos similares del mismo fabricante de *software*, ya que ambos entornos de desarrollo al ser tan completos apenas tienen diferencias importantes que hagan decantarse por uno u otro.

En la Tabla 2 se muestran los principales elementos de ambos entornos.

CONCLUSIÓN

Como se ha podido comprobar a lo largo de este artículo, el mundo *Java* ha crecido a una velocidad impresionante. Hemos pasado de los entornos clásicos con un editor en modo texto a los actuales, donde la variedad permite elegir al usuario lo que más le conviene para realizar su trabajo.

Con la especificación *Java 2* se esperaban entornos muy potentes y completos, y las expectativas se han cumplido. Desde la creación de bases de datos hasta las aplicaciones *CORBA*, se dispone de asistentes y herramientas que facilitan muchas tareas, lo que implica que los desarrollos se realicen en menos tiempo, con mayor calidad y lo que es casi más importante, cuentan con una garantía de reutilización y mantenimiento que hasta ahora era casi imposible de esperar.



MP3 en Linux (y III): Añadiendo reproducción MP3 a nuestros programas (II)

Vicente A. Sánchez Werner.
Desarrollador Independiente.

En el artículo anterior describimos el funcionamiento de una librería que permitía incluir reproducción *MP3* en nuestros programas. En esta ocasión hablaremos de cómo utilizar *SDL* y *SMPEG* para conseguir el mismo propósito.

INTRODUCCIÓN

Visto el uso de la librería *SkySound* para incluir capacidades de reproducción *MP3* en nuestros programas, en este número no hablaremos de una única librería, sino de cómo podemos incluir la reproducción de archivos *MP3* en nuestros programas, utilizando de forma conjunta *SDL* y *SMPEG*, e incorporar nuevas prestaciones obteniendo las ventajas de una arquitectura multihilo.

Antes de iniciar nuestro camino, debemos saber qué son estas librerías y qué funciones van a permitir realizar. *SDL* es el acrónimo de *Simple Direct Media Layer*, una librería *LGPL* desarrollada por *Sam Lantaniga* con el apoyo de *Loki Software* que ofrece una *API* cómoda, sencilla y portátil para la realización de aplicaciones gráficas como son los videojuegos.

La combinación de SDL y SMPEG ofrece muchas más prestaciones que la librería SkySound

Dada su extensa naturaleza, sólo hablaremos de *SDL* en los aspectos concernientes al sonido, recomendando a aquéllos que deseen adentrarse más profundamente en ella, la lectura de la serie de artículos sobre *SDL* que se ha iniciado en el número 17 de nuestra revista hermana, *Sólo Programadores Linux*.

SMPEG es una librería -también bajo licencia *LGPL*- que permite la descodificación de los archivos *MP3* que vayamos a usar, independientemente incluso de la presencia de *SDL*, utilizando funciones propias que descodifican el archivo a un *buffer* para que realicemos con él las más diversas operaciones.

INSTALACIÓN

Estas librerías se pueden obtener fácilmente en la *Red*, o cogién-



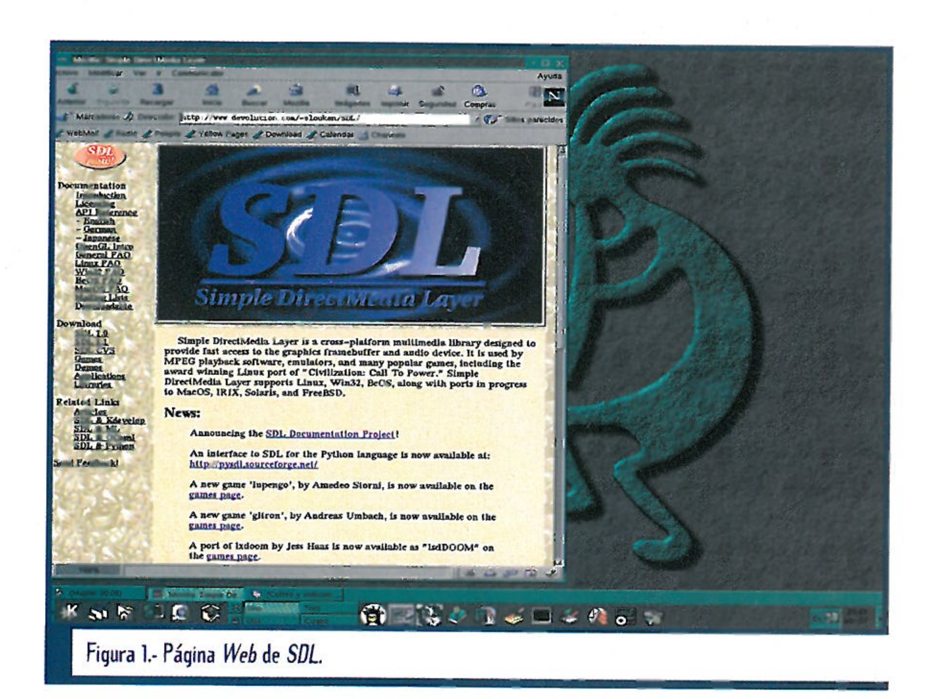


Figura 1.- Página *Web* de *SDL*.

dolas del *CD-ROM* que acompaña a esta revista. En cualquier caso, es conveniente consultar de vez en cuando las páginas *Web* de estas librerías, por si hubiese aparecido una versión más reciente. Las *URL* de las páginas donde podemos encontrarlas son:

- *SDL*. *http://www.devolution.com/~slouken/SDL*. Aquí también hallaremos punteros hacia programas desarrollados con esta librería, y hacia librerías que complementan a *SDL* como *SDL_Mixer*.
- *SMPEG*. *http://www.lokigames.com/developer/smpeg.php3*. En esta página podremos encontrar la librería en varios formatos que facilitan su instalación en las diferentes distribuciones. Además, ofrece un enlace con el grupo de noticias hospedado en *Loki Games* acerca del uso de la misma, así como vínculos a otras librerías y programas *Open Source* desarrollados por esta empresa.

En el *CD-ROM* de este mes hemos incluido la versión *1.1.2* de *SDL* (última de la rama de desarrollo hasta la fecha) y *0.3.5* de *SMPEG*. Éstas son las versiones más actualizadas en el momento de escribir este artículo, aunque es probable que en el momento de su publicación, *SDL* se haya actualizado.

Las librerías SDL y LPG han sido utilizadas con éxito por Loki Games para portar videojuegos de Windows a Linux

Las librerías pueden descargarse de las mencionadas direcciones en varios formatos, aunque el más recomendable siempre es el paquete binario adecuado a la distribución que estemos utilizando, ya que así se instalará de forma perfecta en nuestro sistema. En caso contrario, deberemos bajar los ficheros *tar.gz*, compilarlos e instalarlos siguiendo las instrucciones aparecidas en los mismos.

Una vez que disponemos de las librerías en nuestra máquina, ya podemos comenzar a adentrarnos en su interior.

SDL-1.1.2

La librería *SDL* será la que nos dará el marco principal para trabajar con *SMPEG*, con funciones para la inicialización de los gráficos, sonido, tipos de datos y estructuras que permitirán realizar las operaciones que deseemos de forma cómoda y sencilla.

Para utilizar la librería en nuestros programas deberemos incluir la siguiente línea de código, que indica al compilador dónde buscar el fichero de cabecera *SDL.h*:

```
#include <SDL/SDL.h>
```

En el caso de que utilicemos *Mandrake 7.0* o cualquiera de las derivadas de *RedHat*, esto seguirá siendo válido, pero en otras distribuciones la línea puede ser diferente, dependiendo del lugar donde se haya instalado la librería. Además de esto, deberemos incluir las órdenes necesarias para que en la fase de enlazado del programa, se enlace con las librerías que utilicemos mediante las siguientes opciones del compilador:

```
gcc archivo.c -lpthread -
    L/usr/lib -lSDL
```

Una vez instalada la librería podemos comenzar a utilizarla. En primer lugar deberemos inicializar la librería, esto se consigue mediante la siguiente llamada a la función:

```
Sdl_Init(Banderas)
```

Las banderas pueden ser *SDL_Init_Audio*, o *SDL_Init_Video*. La primera inicializa el sistema de sonido de la librería y la segunda inicializa el sistema de vídeo de la misma. Ejemplos de su uso son:

```
SDL_Init(SDL_Init_Audio);
//Inicializa sólo el sonido
```

```
SDL_Init(SDL_Init_Video);
//Inicializa sólo el vídeo
SDL_Init(SDL_Init_Video|SDL_Init_A
    udio);
//Activa ambos
```

Es también imprescindible conocer la forma de desactivar la librería al salir, ya que si no, podrían quedarse activos los hilos de ejecución creados por la librería. Esto se consigue mediante *SDL_Quit()*. Hemos de tener en cuenta que esta llamada no libera la memoria que hemos utilizado, por lo que antes de realizar un *SDL_Quit*, deberíamos liberar la memoria dinámica que antes hemos consumido.

SDL es mucho más extensa, y dispone de su propia *API* básica de sonido. Pero en nuestro caso, estas llamadas sólo son una complicación adicional, ya que *SMPEG* las usa en un nivel más cercano a nosotros, disminuyendo su dificultad de uso. Lo que sí nos interesa es la forma de poder seleccionar un dispositivo de salida determinado para la reproducción de audio.

La selección del dispositivo de salida puede realizarse entre los siguientes posibles:

- *dsp* (Usado por defecto). Este dispositivo utiliza la *API OSS* para abrir y trabajar en el dispositivo */dev/dsp*.
- *dma*. Utiliza la *API OSS* para realizar accesos a través de *DMA* al dispositivo */dev/dsp*
- *esd*. Utiliza el demonio *ESound* para emitir el sonido.

La posibilidad de optar por uno o varios de estos dispositivos depende de las opciones con que hayamos compilado la librería *SDL*. La selección de uno u otro se realiza mediante la utilización de la variable de entorno *SDL_AUDIODRIVER*,

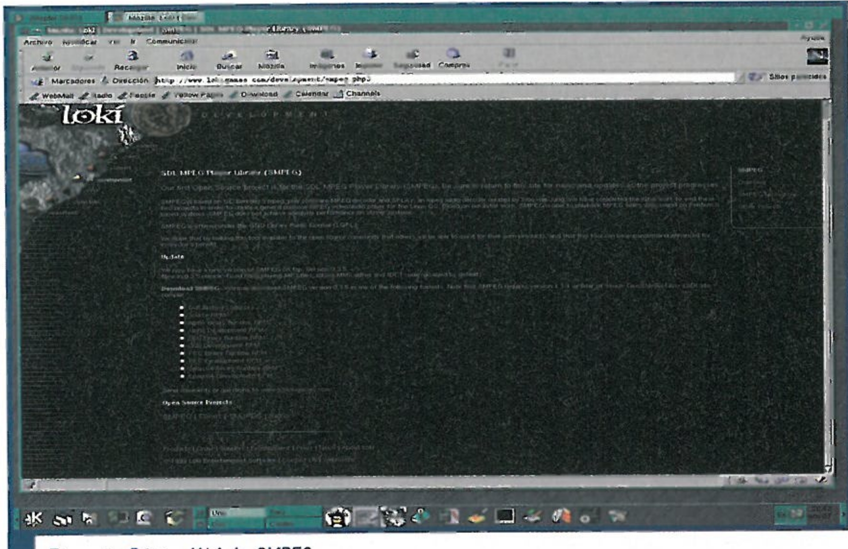


Figura 2.- Página *Web* de *SMPEG*.

que contendrá el nombre del dispositivo de salida que vamos a usar.

SDL ofrece funciones de vídeo y sonido para el desarrollo de aplicaciones portables

Una vez que ya tenemos unas mínimas nociones de cómo inicializar la librería para utilizar sus funciones de sonido, podemos adentrarnos en *SMPEG* y comenzar a reproducir *MP3*.

SMPEG-0.3.5

La librería *SMPEG* ha sido desarrollada por *Loki Games* para ser utilizada en el desarrollo de sus videojuegos, y posee capacidad, no sólo para reproducir flujos *MP3*, sino para visualizar archivos de vídeo *MPEG-1*. Sin embargo, antes de lanzarnos alocadamente sobre ella, debemos tener en cuenta un importante factor: la velocidad. La librería es bastante lenta, y eso supone que en ordenadores inferiores al *Pentium II* no pueda reproducir archivos *MP3* con la máxima calidad y la total certeza de que no se sufrirán cortes o parones en la reproducción.

Una vez que nos hemos decidido a utilizar esta librería, deberemos instalarla, bien utilizando un paquete binario, o bien mediante la compilación del código fuente. Es importante que tengamos correctamente instalada la librería *SDL 1.1.1* como mínimo, de lo contrario la compilación fallará y *SMPEG-0.3.5* no se podrá instalar. Con la librería ya instalada, debemos incluir en nuestros programas una línea similar a la siguiente:

```
#include <SMPEG/SMPEG.h>
```

Y al igual que en el caso anterior, deberemos modificar las opciones del compilador, para que nuestros programas se enlacen con esta librería. Por ejemplo:

```
gcc archivo.c -lpthread -
    L/usr/lib -lSDL -lsmpeg
```

Una vez que hemos configurado adecuadamente la librería en nuestro sistema, podemos comenzar a integrar la reproducción de archi-



vos *MP3* en nuestros programas. Para esto, el primer paso consiste en conocer los tipos de datos que permiten realizar esta función en el interior de nuestros programas.

Las llamadas de SDL más interesantes son las de inicialización y salida

En el nivel básico en el que nos estamos moviendo, existen dos tipos de datos que siempre usaremos cuando integremos la reproducción de *MP3* en un programa dado. La primera de estas estructuras es *SMPEG*, que representa al archivo que deseamos reproducir. Dado el uso que vamos a hacer de ella, lo más habitual es que las variables de este tipo se declaren como punteros y, dada la naturaleza de los mismos, es necesario que antes de usar estas variables las inicialicemos a *NULL*, con el fin de evitar problemas. Un ejemplo de declaración sería:

```
SMPEG *musica;
```

El otro tipo que vamos a necesitar es *SMPEG_Info*, que es una estructura que contiene las características de reproducción del fichero que vamos a reproducir. Estas características permiten conocer cómo se va a reproducir el archivo. Habitualmente las variables de este tipo de datos van a ser declaradas normalmente, y no como punteros. Un ejemplo podría ser:

```
SMPEG_Info informacion;
```

El siguiente paso consiste en indicar a la librería qué archivo es el elegido para ser reproducido. Esto se realiza mediante la siguiente instrucción:

```
musica=SMPEG_new("test.mp3",&infor
     macion,1);
```

Esta instrucción llama a la función *SMPEG_New* y le dice que coloque en la variable *musica* toda la información necesaria para reproducir el archivo *test.mp3,* y que, además, deje en *informacion* las características de reproducción de este archivo. El último parámetro indica que se usará el sistema de sonido de *SDL* para llevarla a cabo. Es importante hacer notar que esto requiere que la inicialización de los subsistemas de vídeo y sonido

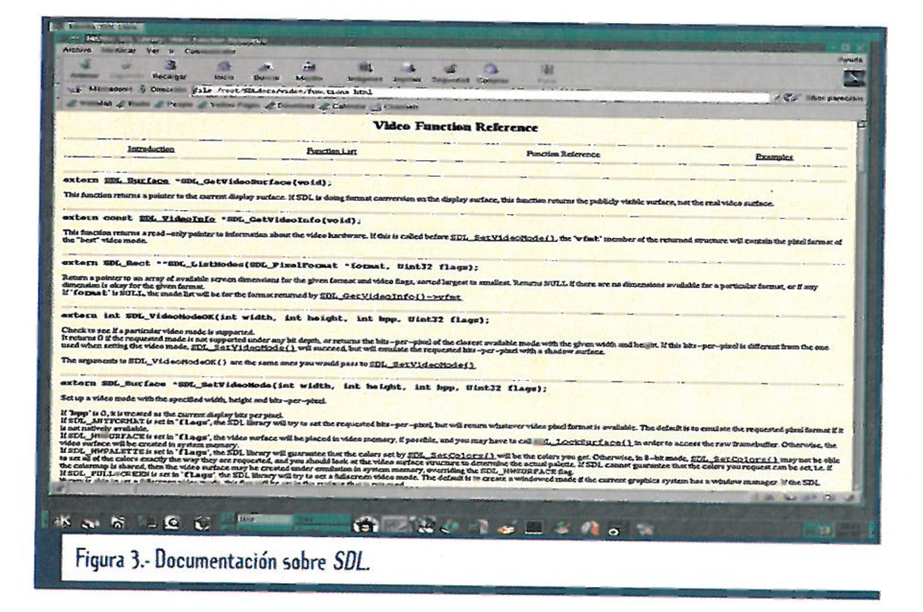
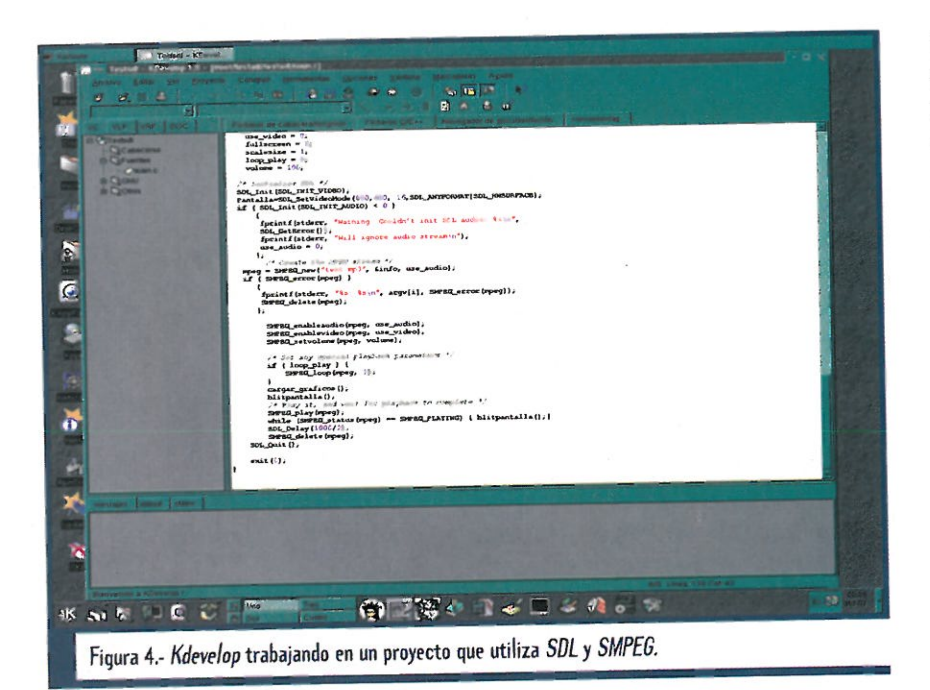


Figura 3.- Documentación sobre *SDL*.


Figura 4.- *Kdevelop* trabajando en un proyecto que utiliza *SDL* y *SMPEG*.

de *SDL* se realice en dos órdenes diferentes, y no de forma conjunta, ya que *SMPEG* creará su propia conexión al subsistema de sonido, cosa imposible si la inicialización se hubiese realizado de forma conjunta. En caso de indicar un *0*, deberemos ser nosotros los que creemos la conexión de *SMPEG* con el sistema de sonido de *SDL*, lo que requiere el uso de la *API* de sonido de *SDL* y eleva la dificultad de uso.

SMPEG, además de reproducir ficheros de sonido MP3, también descodifica archivos MPEG-1 de sonido y vídeo

Esta operación puede fallar, y si no sabemos qué está fallando, la aplicación producirá un error de segmentación al intentar reproducir algo que no existe, terminando de forma brusca con nuestras ilusiones. La forma de evitar esta embarazosa situación consiste en utilizar la función que proporciona *SMPEG* para comprobar que las operaciones se llevan a cabo con éxito. Esta función es: *SMPEG_error*, y tomando como parámetro una variable de tipo *SMPEG ** nos indica si se ha encontrado algún problema que pueda interrumpir su reproducción. La forma más habitual de realizar la comprobación puede verse a continuación:

```
mpeg = SMPEG_new("test.mp3",
     &info, use_audio);
if ( SMPEG_error(mpeg) )
{printf("%s\n", SMPEG_error(mpeg))
  SMPEG_delete(mpeg);
};
```

En este breve código, primero indicamos a la librería qué archivo queremos preparar para su posterior reproducción, y después comprobamos que no haya ningún error. En el momento en que se produzca (por ejemplo, archivo *test.mp3* inexistente o corrupto), se emite un mensaje de error de la librería gracias a la instrucción *printf("%s\n", SMPEG_error(mpeg))*, y se libera toda la memoria que hubiese podido ser reservada para la reproducción del archivo mediante la instrucción *SMPEG_delete(mpeg)*. Esta última función es importante para poder terminar la ejecución de la aplicación de la mejor forma posible, liberando toda aquella memoria que haya podido ser reservada para nuestro uso.

Antes de iniciar la reproducción debemos realizar algunas comprobaciones, de cara a no consumir más recursos de los que serían necesarios para nuestro propósito. Las características se pueden consultar a través de una variable de tipo *SMPEG_info* que haya sido usada con la función *SMPEG_new* para preparar el fichero. Los campos más interesantes de que dispone este tipo de datos son los siguientes:

- *has_audio*. Este campo vale 1 si el archivo dispone de algún canal de sonido.
- *has_video*. Idéntico al anterior, pero referido exclusivamente a los canales de vídeo.
- *width*. Anchura en píxeles del canal de vídeo.
- *Height*. Altura en píxeles del canal de vídeo.

En una aplicación podríamos realizar la consulta de estos campos de la siguiente forma:

```
mpeg = SMPEG_new("test.mp3",
    &info, use_audio);
if (info.has_audio) {
  printf("Tiene Sonido\n");
}
else {
  printf("No tiene sonido|n");
};
if (info.has_video) {
 printf("Tiene video\n");}
else {
 printf("No tiene video|n");
};
```

Conocer estas prestaciones nos permite indicar a la librería que no descodifique el sonido o el vídeo, por lo que la reproducción requerirá menos recursos y será más ágil. Esto se realiza mediante las siguientes funciones:



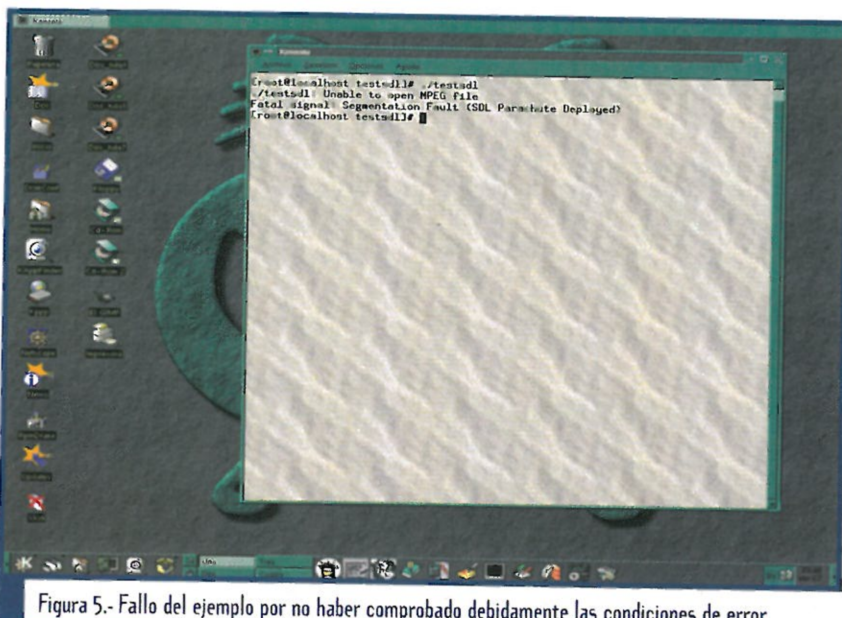


Figura 5.- Fallo del ejemplo por no haber comprobado debidamente las condiciones de error.

- *SMPEG_enableaudio(SMPEG *, numero)*. Esta función permite habilitar o deshabilitar la descodificación del sonido en un archivo determinado indicando un 1 o un 0 respectivamente en *numero*.
- *SMPEG_enablevideo(SMPEG *, numero)*. Esta función permite habilitar o deshabilitar la descodificación del vídeo en un archivo determinado, indicando de nuevo un 1 o un 0 respectivamente en *numero*.

Con estas dos funciones, optimizaremos la carga del sistema al descodificar los archivos, realizando únicamente la descodificación del canal que interese.

El volumen del sonido generado puede establecerse desde este mismo momento, aunque también puede modificarse mientras el archivo está reproduciéndose. El ajuste de volumen se realiza mediante la función *SMPEG_setvolume* que tiene la siguiente sintaxis:

Para alcanzar la mayor calidad de reproducción con SMPEG, es necesario un equipo no inferior a un Pentium II

- *SMPEG_setvolume(SMPEG *, volumen)*. Esta función selecciona el volumen de reproducción del archivo indicado mediante la variable *SMPEG **. El volumen será un número entre 0 y 100.

Antes de empezar la reproducción propiamente dicha, podemos indicar a la librería que reproduzca el archivo una única vez, o que lo

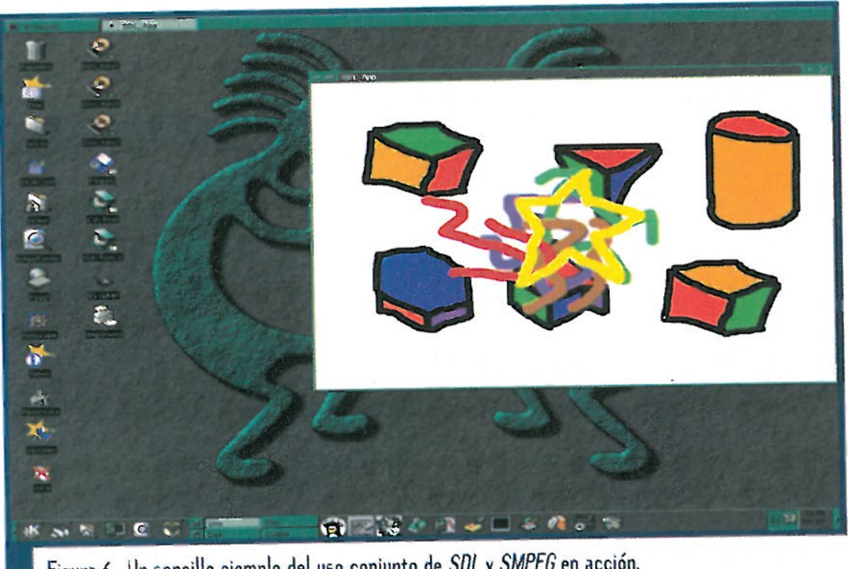


Figura 6.- Un sencillo ejemplo del uso conjunto de *SDL* y *SMPEG* en acción.

reproduzca de forma continua hasta que se le indique lo contrario mediante la siguiente función:

- *SMPEG_loop(SMPEG *, numero)*. Esta función decide si el archivo será reproducido de forma continua o no. En el primer caso en número colocaremos un 1 y en el segundo caso colocaremos un 0.

SMPEG se instala con paquete binario, o por compilación de código fuente

Finalmente, ya hemos llegado al punto en que podemos realizar la reproducción del archivo notándose una gran diferencia de comportamiento respecto a la librería *SkySound* tratada en el número anterior. La reproducción del archivo se inicia mediante la función *SMPEG_Play(SMPEG *)*. Ésta inicia la reproducción del archivo por un hilo de ejecución secundario por lo que no es necesario que se realice el *"polling"* que requería la librería *SkySound*, pudiendo realizar otras tareas simultáneamente, como muestra el ejemplo incluido en el *CD-ROM* que acompaña a la revista de este mes.

Sin embargo, sí es interesante que podamos comprobar si el archivo se está reproduciendo o no en un momento dado, para lo que utilizamos la función *SMPEG _status(SMPEG *)* que devuelve el estado de la reproducción del archivo indicado por *SMPEG **. Este estado puede ser uno de los siguientes:

- *SMPEG_ERROR*. Ha habido algún error durante la reproducción.
- *SMPEG_STOPPED*. La reproducción ha cesado.
- *SMPEG_PLAYING*. El archivo está reproduciéndose.

Esto permite realizar acciones según el estado de la reproducción, o esperar a que un archivo termine de reproducirse.

Debemos añadir *SMPEG* brinda la oportunidad de pausar, parar o reiniciar la reproducción del archivo en cuestión mediante las siguientes funciones:

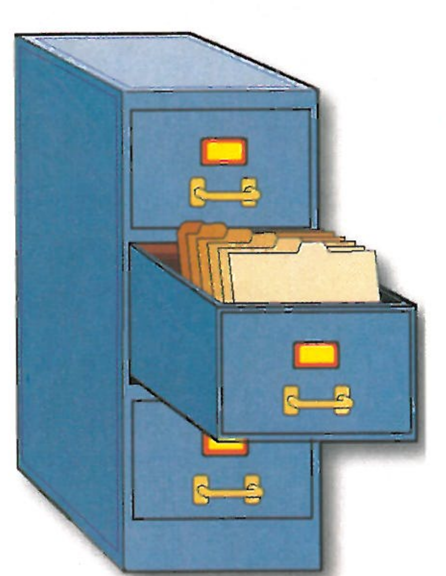
- *SMPEG_pause(SMPEG *)*. Esta función para o retoma la reproducción del archivo.
- *SMPEG_stop(SMPEG *)*. Esta función interrumpe la reproducción del archivo en cuestión.
- *SMPEG_rewind(SMPEG* mpeg)*. Reproduce el archivo desde su inicio.

Finalmente, debemos mencionar que *SMPEG* dispone de varias llamadas adicionales que permiten controlar la reproducción del archivo a un nivel más bajo, gestionar la descodificación del vídeo, desplazarnos a través del mismo, o descodificar a un *buffer*. Sin embargo, estas últimas se salen del objetivo del artículo.

CONCLUSIÓN

Este artículo pretendía mostrar una opción alternativa a *Sky Sound* para la descodificación de archivos *MP3* y mostrar parte de la potencia de las librerías *SDL* y *SMPEG*. En él, hemos podido ver cómo se pueden realizar casi las mismas operaciones que utilizando *SkySound*, pero no todas, ya que para ello es necesaria la intervención de una tercera librería llamada *SDL_mixer* que expande aún más las posibilidades de esta pareja, añadiendo, por ejemplo, la mezcla de sonido mediante varios canales. En cualquier caso, se puede entrever la mayor potencia de la combinación *SDL-SMPEG*, y queda abierto el camino a los que deseen explorar el tema más profundamente.

Terminamos aquí esta serie sobre la inclusión de tecnologías *MP3* en aplicaciones *Linux*, con la confianza de haber abierto la puerta a nuevas posibilidades para aquellos programadores que desconocían estas funcionalidades, o la forma de utilizarlas.



Aplicaciones de bases de datos con Delphi 5 (IV)

Juan Luis Ceada Ramos.
Diplomado en Informática de Sistemas.

En esta entrega veremos la forma de trabajar con diversos sistemas gestores de bases de datos *(SGBD's)*, nuevos componentes de la *VCL*, así como algunas nociones de *SQL*.

SISTEMAS GESTORES DE BBDD

Un sistema gestor de bases de datos no es más que una aplicación que se encarga, como su propio nombre indica, de gestionar el acceso y manipulación de los datos que tiene a su cargo. Ejemplos comerciales de gestores de bases de datos son *Oracle*, *Interbase*, *SQL Server*, etc.

Estos gestores se apoyan (en la mayoría de los casos) en la arquitectura cliente/servidor y en el lenguaje *SQL*. Los programas clientes (en este caso, desarrollados en *Delphi*), no acceden directamente a los datos. En su lugar, cualquier petición (ya sean consultas o modificaciones) se pasan al *SGBD*, que suele estar situado en otra máquina (mucho más potente). Dichas peticiones no son más que consultas *SQL*, que el *SGBD* interpreta y ejecuta, devolviendo tras el proceso los resultados al cliente.

La comunicación entre el cliente y el servidor puede realizarse mediante múltiples protocolos, siendo *TCP/IP* uno de los más extendidos. Las ventajas de este sistema son numerosas: mayor control sobre los accesos concurrentes,

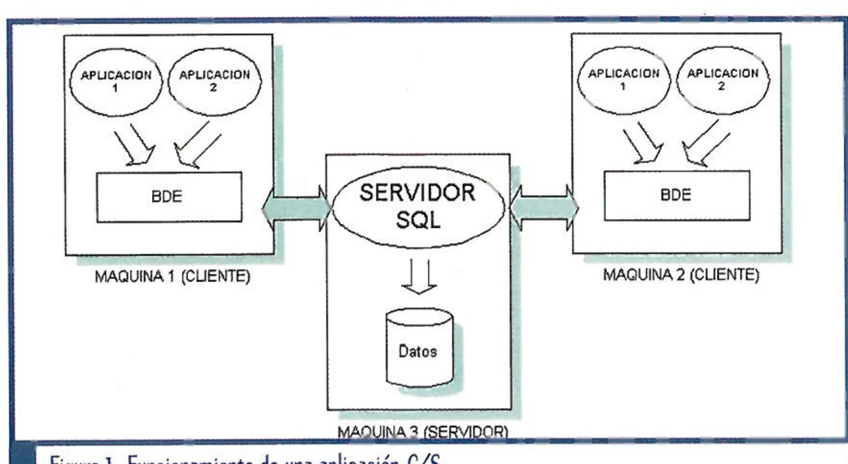


Figura 1.- Funcionamiento de una aplicación *C/S*.



posibilidad de usar transacciones, posibilidad de programar el servidor, posibilidad de realizar operaciones complejas con los datos, etc.

Los SGBD's permiten trabajar usando transacciones

El hecho de que todos los clientes tengan que pasar por el servidor para poder acceder a los datos (ya sea para leer o escribir), hace que se controlen con más precisión los accesos concurrentes a las diferentes bases de datos. El servidor se encarga, en función de las operaciones que realicen los usuarios, de bloquear/liberar los registros implicados en las operaciones, de forma que nunca haya dos usuarios editando el mismo registro a la vez (o mejor dicho, sí pueden, pero el segundo que intente grabar los datos recibirá un error como respuesta).

Los *SGBD* también permiten definir transacciones. Una transacción es una operación atómica. Una vez que el usuario da comienzo a una transacción, puede realizar todas las acciones que desee sobre la base de datos: borrar, añadir registros, editar, crear tablas, etc. En cualquier momento, el usuario puede confirmar todas las operaciones que haya realizado, mediante un *COMMIT*. En ese instante, todo lo que haya realizado desde que abrió la transacción se graba físicamente en la base de datos.

Si alguna de las operaciones provoca un error, basta con efectuar una operación *ROLLBACK* para anular todos los cambios desde que dio comienzo la transacción.

En pocas palabras, una vez comience la transacción, sólo quedan dos posibilidades: aceptar todo, o cancelarlo todo. Esto es tremendamente útil en determinados casos, porque en caso de fallos (por ejemplo, un corte del suministro eléctrico), se garantiza que las bases de datos van a quedar en un estado consistente.

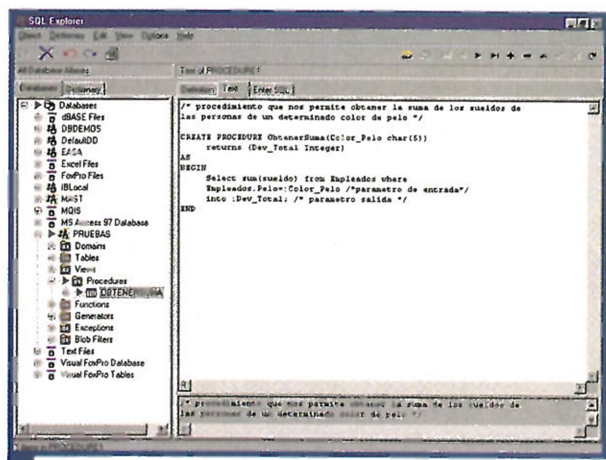


Figura 2.- Creando un procedimiento almacenado en una *BD Interbase*.

Por mencionar otra ventaja más, citaremos el hecho de que es posible programar determinadas operaciones en el *SGBD*, liberando de estas tareas a los clientes, reduciendo el tráfico de la red, y aumentando la velocidad del sistema. Para ello, existen diversos mecanismos, como los procedimientos almacenados y *triggers*. Hay que indicar que los *SGBD's* de "escritorio", como *Access*, no soportan la programación en el lado del servidor, por lo tanto, no admiten procedimientos ni *triggers*.

Un procedimiento almacenado no es más que un conjunto de sentencias almacenadas en el servidor que el cliente puede ejecutar cuando lo desee. En general, es altamente recomendable que cualquier procedimiento que implique trabajo intensivo con los datos, y que no requiera de la interacción con el usuario, sea transformado en un procedimiento almacenado.

Un *trigger* está formado por un conjunto de sentencias que se ejecutan en determinadas circunstancias. Por ejemplo, podríamos tener un *trigger* que saltase justo antes de borrar un registro, que se encargase de eliminar los registros asociados (en una relación uno a muchos) al registro que estamos borrando.

Procedimientos almacenados y triggers permiten aprovechar la potencia de los SGBD

Uno de los inconvenientes de usar *SGBD's* (a parte del encarecimiento del producto final) es que, en algunos casos, las aplicaciones funcionarán más lentamente que sus equivalentes que trabajen con *BBDD* locales (tipo *Paradox*), aunque esto se puede paliar en gran medida pasando parte de la programación al servidor (como hemos comentado en líneas anteriores).

Delphi proporciona los mecanismos apropiados para poder trabajar y explotar al máximo las características de los *SGBD's*, gracias a los componentes *TDatabase*, *TStored Proc*, *TQuery*, etc.



Figura 3.- Algunos de los componentes que permiten aprovechar la potencia de los *SGBD's*.

NOCIONES DE SQL

El lenguaje *SQL* es un lenguaje de definición y manipulación de datos. En teoría se ha diseñado para permitir a cualquier usuario (no necesariamente informático) acceder a los datos contenidos en servidores *SQL* de una forma rápida y sencilla. Suponiendo que las tablas de una base de datos se encuentren en tercera forma normal, podemos asegurar que mediante instrucciones *SQL*, podemos acceder al más recóndito de los campos.

SQL es un lenguaje estándar que incorporan todos los SGBD's

Actualmente, *SQL* es un lenguaje estándar que implementan todos los sistemas gestores de bases de datos relacionales. Si bien es cierto que no todos los gestores lo implementan de la misma forma, sí podemos estar seguros de que todos cumplen casi al 100% con el estándar *SQL*-92, aprobado por organismos como *ISO* o *ANSI*.

Las instrucciones *SQL* se pueden agrupar en dos categorías: instrucciones para el manejo de estructuras de datos (*DDL*) e instrucciones para el manejo de los datos *DML*.

Las instrucciones *DDL* permiten crear tablas, borrarlas, añadir índices, nuevos campos, crear vistas de los datos, etc. No las veremos en esta entrega, puesto que podemos hacer todas estas operaciones usando entornos gráficos, como el propio *SQL Explorer* que incluye *Delphi*.

Las instrucciones *DML* permiten realizar consultas sobre los datos, insertar registros, eliminar registros, etc. A continuación veremos cuáles son las principales instrucciones *DML*. Puesto que esto no pretende ser un curso de *SQL*, nos limitaremos a dar los conceptos básicos, acompañados de algunos ejemplos. En *www.todo-gratis.com* podréis encontrar cursos de *SQL* de diferentes niveles.

SENTENCIA SELECT

La sentencia *Select* es la madre de todas las sentencias, siendo con diferencia la más usada, y por tanto, su dominio resulta imprescindible. Esta es su estructura básica (en notación *BNF*):

```
Select [distinct] lista-
     expresiones
From lista-tablas
Where lista-condiciones
Order by lista-columnas
```

La cláusula *Select*, seguida de una lista de expresiones, permite especificar qué campos de la consulta queremos obtener. También podemos hacer operaciones con un determinado campo (como por ejemplo, obtener la suma del campo cantidad de todos los registros de una tabla). Si añadimos *distinct*, se eliminan todos los registros duplicados que aparezcan en el resultado.

La cláusula *From* permite especificar de qué tablas se extraen los datos (una, o varias relacionadas entre sí). La cláusula *Where* permite indicar qué registros se incluirán en el resultado, mediante los operadores lógicos usuales. Por último, la cláusula *order by* permite especificar un criterio de ordenación para los resultados obtenidos, por cualquiera de los campos solicitados en la cláusula *Select*.

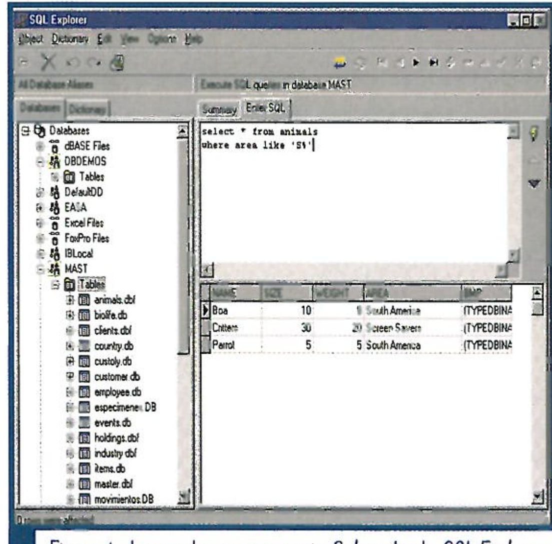


Figura 4.- Lanzando una sentencia *Select* desde *SQL Explorer*.

Ejemplos:

1. Obtener todos los registros de una tabla.

```
Select * from Clientes
```

2. Obtener todos los clientes de una determinada localidad.

```
Select * from Clientes where
     localidad='Huelva'
```

3. Obtener todos los empleados que cobran entre 100-200.000.

```
Select * from Empleados
where Sueldo between 100000 and
     200000
```

4. La 2 y la 3 combinadas.

```
Select * from Empleados
where (Sueldo between 100000 and
     200000) and
     (localidad='Huelva')
```

5. Obtener todos los artículos de una factura.

```
Select Facturas.Numero,
     Articulos.Descripcion
from Facturas, Articulos
where Facturas.Numero=Articulos.
     Numero
```



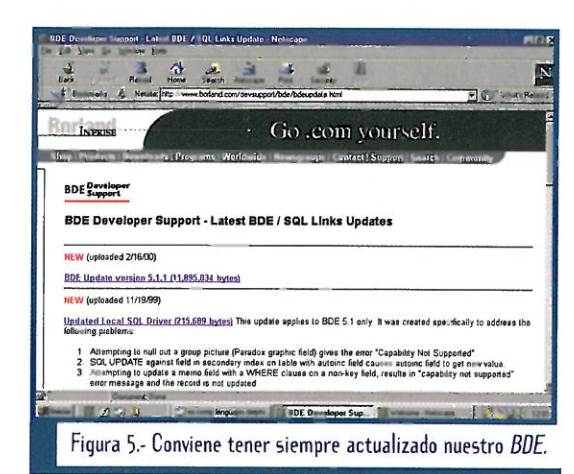


Figura 5.- Conviene tener siempre actualizado nuestro *BDE*.

6. Obtener la suma de los sueldos de todos los empleados.

```
Select sum(sueldo) from Empleados
```

SENTENCIA INSERT

Mediante esta sentencia podremos añadir nuevos registros a la base de datos. A continuación mostramos su estructura y algunos ejemplos de uso:

```
INSERT INTO tabla (columnas)
(VALUES (valores)) | (Sentencia
     Select)
```

1. Insertar un cliente en la tabla .

```
Insert into Clientes (nombre,
     dni, telefono)
          values
('Pedro', 44123456,
'999-99-99-99')
```

2. Insertar los clientes en la tabla morosos.

```
Insert into Morosos
(nombre, dni, telefono)
     Select Nombre, dni,
telefono from clientes
     where saldo<0
```

SENTENCIA UPDATE

Con ella actualizaremos registros en la *BBDD*. Mostramos su estructura y ejemplos de uso:

```
Update tabla Set columna=valor [,
     columna=valor....]
Where lista-condiciones
```

1. Igualar el sueldo de todos los empleados a 150.000.

```
Update Empleados Set
     Sueldo=150000
```

2. Convertir en morenos y subir el sueldo a todos los que sean rubios.

```
Update Empleados Set
     Pelo='Moreno', Sueldo=200000
Where Pelo='Rubio'
```

SENTENCIA DELETE

Con esta sentencia podremos eliminar determinados registros en la base de datos. A continuación mostramos su estructura y algunos ejemplos de uso:

```
Delete From tabla
Where lista-condiciones
```

1. Eliminar a todos los empleados.

```
Delete From Empleados
```

2. Eliminar sólo a los casados

```
Delete From Empleados
   where Estado='C'
```

¿SQL IMPLICA SGBD?

No. No es necesario contar con un *SGBD* para poder usar el lenguaje *SQL*. Gracias al *BDE* podemos usar el lenguaje *SQL* para consultar y modificar datos de cualquier base de datos a la que el *BDE* proporcione acceso (como por ejemplo, *Paradox o Access*). El *BDE* se encarga de procesar la consulta *SQL* si es necesario. Por supuesto, el motor de *SQL* del *BDE* tiene sus limitaciones, y tampoco podemos pedir peras al olmo.

TABLA 1. Principales propiedades y métodos de TDatabase (P=propiedad, M=método)

Nombre	Tipo	Descripción
AliasName	P	Indica el alias que usaremos para acceder a la base de datos deseada.
Connected	P	Si vale **True**, quiere decir que hemos conectado con la base de datos deseada.
DatabaseName	P	Si especificamos un valor, éste se convierte en un alias local, que debe ser usado por el resto de los componentes, en lugar del alias global declarado en *AliasName*.
InTransaction	P	Indica si hay una transacción en Progreso.
IsSQLBased	P	Si vale **True**, la conexión se ha realizado mediante *ODBC* o mediante un *SQL Link*.
LoginPrompt	P	Si vale **True**, al conectar con la *BD*, se muestra un cuadro de diálogo para que el usuario se identifique, ignorando los parámetros introducidos mediante *Params*.
Params	P	Contiene una serie de parámetros que se pasan a la base de datos a la hora de conectar (como por ejemplo, usuario y *password*).
Commit	M	Confirma la transacción activa.
Rollback	M	Cancela la transacción activa.
StartTransaction	M	Da comienzo una nueva transacción. No se permiten anidar transacciones.

TABLA 2. Cómo configurar un TDatabase para acceder a una base de datos.

Propiedad	Valor
AliasName	MiBDInterbase (nombre del alias que apunta a la *BD Interbase*).
DatabaseName	MiBDLocal (alias que usarán el resto de los componentes).
LoginPrompt	False
Params	Ver Figura 6.
Connect	True

¿ES NECESARIO EL USO DE SQL?

No siempre. Si nuestra aplicación se limita a insertar datos, no necesitamos usar *SQL* para nada. Bastaría con incluir un *TTable*, enlazarlo a una tabla, y usar los métodos *Insert*, *Append*, etc., que vimos en pasadas entregas. El *BDE* se encarga de transformar estas llamadas a métodos en sentencias *SQL*.

El componente TDataBase permite trabajar con transacciones

Si nuestras aplicaciones realizan consultas complejas sobre la base de datos, casi con toda seguridad nos veremos obligados a usar sentencias *Select*, que en combinación con los *TQuerys*, permitan acceder a los datos.

En definitiva, el uso de *SQL* en nuestras aplicaciones se limitará, en la mayoría de los casos, a consultas sobre los datos, y algún que otro uso esporádico de los *insert/update/delete*. Eso suponiendo que la edición de datos la hagamos mediante componentes de acceso a datos, y no sobre controles estándar (es decir, sobre *TDBEdits* y no *TEdits*), en cuyo caso todas las operaciones con la *BD* se deberían hacer mediante *SQL*.

NUEVOS COMPONENTES

Veamos ahora algunos componentes que serán útiles a la hora de trabajar con *SGBD's*. Ya vimos en su momento el componente *TQuery*, que permitirá ejecutar cualquier sentencia *SQL* que se nos antoje, sin más que asignársela a su propiedad *SQL*, para después llamar al método *Open*. Ahora veremos un par de componentes más.

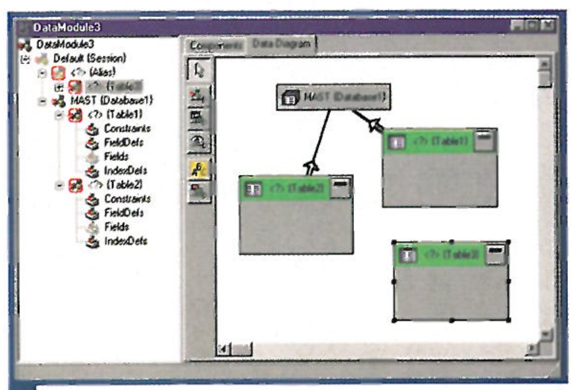


Figura 6.- Un *TDatabase*, con dos *TTables* asociados que usan el alias local y otro que no lo hace.

TDATABASE

El componente *TDatabase* permite conectar con bases de datos remotas, enviando, cuando sea necesario, el nombre de usuario y la *password* de éste al servidor *SQL*. Además, permitirá definir alias locales (es decir, sólo están activos durante la ejecución de la aplicación), así como controlar las transacciones.

En nuestro módulo de datos bastará con incluir un *TDatabase* por cada base de datos a la que queramos acceder, y configurar sus principales propiedades, que se muestran en la Tabla 1.

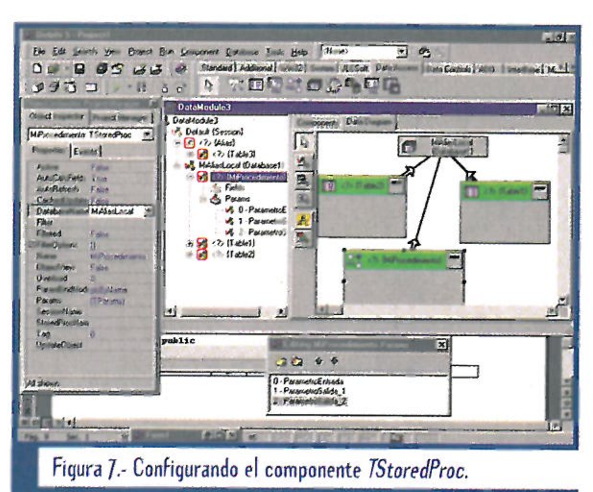
La Tabla 2 muestra cómo configurar un *TDatabase* para que acceda a una base de datos en un servidor *Interbase*.

Una de las principales funciones de *TDatabase* es el control de transacciones. Para ello, proporciona tres métodos: *StarTransaction*, *Commit* y *Rollback*. El primero inicia una nueva transacción. El segundo cierra la transacción, aceptando todos los cambios. El tercer método cancela todas las operaciones que se han lanzado desde que se abrió la transacción, cerrándola a continuación. **No es posible anidar transacciones.**

El siguiente código muestra un pequeño ejemplo del uso de transacciones:

```
try
   if not MiDB.InTransaction then
     MiDB.StartTransaction;
    //Hacer operaciones con la BD
    ....
    //Todo va bien, aceptamos
    MiDB.Commit;
except
    //ha habido problemas
    MiDB.Rollback;
end
```




Figura 7.- Configurando el componente *TStoredProc*.

Como podéis ver, usamos las excepciones para cancelar todas las operaciones en caso de que se produzca algún error durante la manipulación de los datos. Es importante que todos los componentes de acceso a datos (*TTables*, *TQuerys*, etc.) accedan a la *BD* a través del alias local definido por el *TDataBase*, o de lo contrario, no funcionará correctamente el mecanismo de transacciones.

TSTOREDPROC

Mediante este componente podremos acceder a los procedimientos almacenados presentes en la base de datos. Su uso es muy sencillo. Basta con insertar uno en el módulo de datos y configurar un par de propiedades (el nombre de la base de datos, y el nombre del procedimiento almacenado), y llamar al método *ExecProc*. Si el procedimiento recibe parámetros, tendremos que dar valores a dichos parámetros mediante la propiedad *Params*.

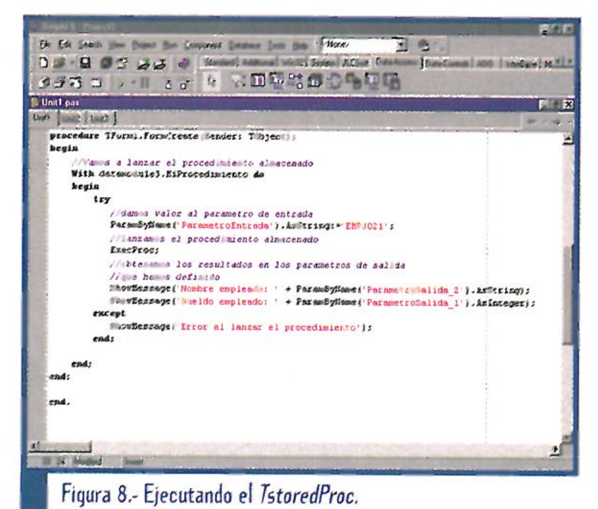


Figura 8.- Ejecutando el *TstoredProc*.

Del mismo modo, un procedimiento almacenado puede devolver resultados en forma de parámetros de salida, a los que se puede acceder mediante la propiedad *Params*. En la Tabla 3 se muestran todas las propiedades de *TStoredProc*. En las Figuras 7 y 8 podéis ver un ejemplo de uso de un procedimiento almacenado.

EL PRÓXIMO MES

En la próxima entrega entraremos de lleno en la parte práctica. Comenzaremos explicando cómo conectar con algunos de los sistemas gestores de bases de datos más comunes (como *Interbase*, *Oracle*, etc.), tanto mediante enlace nativo, como a través de *ODBC*. Aunque no se trate de un *SGBD* propiamente dicho, también veremos cómo conectar con tablas en formato *Access* 97.

Todos los TDatasets deberían usar el alias local definido por el TDataBase

Después haremos una pequeña aplicación donde combinaremos el uso de componentes de bases de datos con sentencias *SQL* embebidas para añadir, editar, o eliminar registros. También veremos para qué sirven las transacciones y accederemos a un procedimiento almacenado. Procuraremos que una misma aplicación sirva para trabajar con diferentes gestores, sin más que cambiar la propiedad *AliasName* del *TDatabase*.

TABLA 3. Principales propiedades y métodos de TStoredProc (p=propiedad, M=método).

Nombre	Tipo	Descripción
DataBaseName	P	Alias de la base de datos (en caso de que haya un *TDataBase*, usaremos el alias local).
Params	P	Parámetros que le pasamos al procedimiento almacenado.
StoredProcName	P	Nombre del procedimiento almacenado en la base de datos.
ExecProc	M	Ejecuta el procedimiento almacenado.
ParamByName	M	Nos permite obtener los resultados que se devuelven como parámetro de salida de los procedimientos.



Programación orientada a objetos (II)

Jordi Agost Moré.
Profesor de programación/Multimedia de la Universidad de Lleida

Después del aperitivo que nos ofreció el artículo anterior sobre las nuevas capacidades de orientación a objetos de *Visual Basic 7*, vamos a ver qué podemos hacer con las posibilidades actuales de *Visual Basic 6*. De momento nos tendremos que conformar con esto hasta que *VB7* esté en la calle.

INTRODUCCIÓN

En este artículo veremos una introducción a la programación de objetos en *VB*, comenzando en primer lugar por las características de los objetos mismos. A continuación abordaremos la tarea de construir clases, así como la manera de trabajar con herencia y polimorfismo.

Visual Basic ha ido entrando poco a poco en el mundo de la programación orientada a objetos, haciendo siempre énfasis en la transición desde la programación por procedimientos. Y más en la versión 6.0 donde a las clases que aparecieron en la versión 4.0 de *Visual Basic*, se le ha añadido ahora el polimorfismo.

Veamos en primer lugar un poco la filosofía y términos de la programación orientada a objetos. Desde un punto de vista general podríamos decir que una clase describe un grupo de objetos similares. Por ejemplo, y moviéndonos dentro del marco empresarial podríamos decir que todos los empleados de una compañía dada son objetos de la clase empleados. Siguiendo con el mismo ejemplo también podríamos

Agregar procedimiento
Nombre:
Aceptar
Cancelar
Tipo
Procedimiento
Propiedad
Función
Evento
Alcance
Público
Privado
Todas las variables locales son estáticas

Figura 1.- Al insertar un nuevo procedimiento se nos permite elegir un procedimiento de clase.



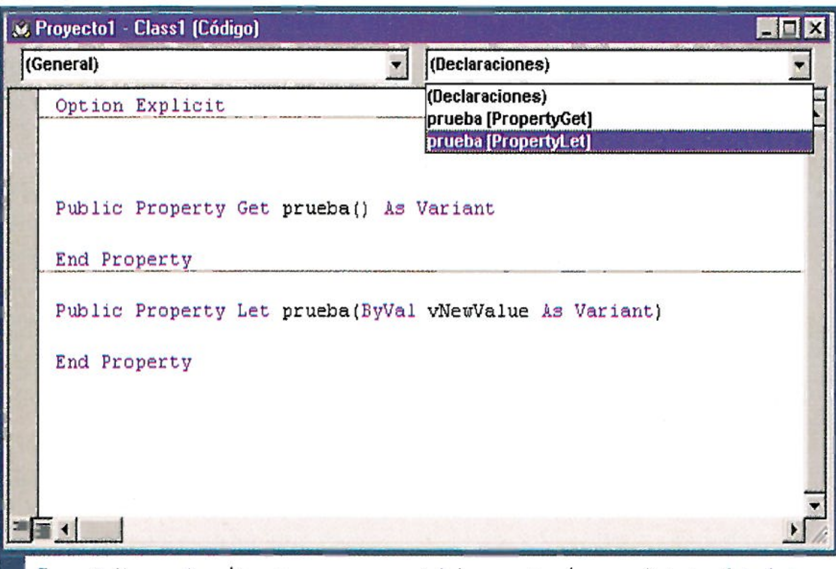


Figura 2.- Vemos cómo al insertar una nueva propiedad se muestran los procedimientos *Get* y *Let*.

decir que todas las delegaciones de dicha compañía son objetos de la clase delegación. Así podemos decir que la *Delegación de Madrid* es una instancia de la clase *Delegaciones* o que el empleado *Pedro González* es una instancia de la clase *Empleados*.

Con Visual Basic 6 se logra dar un nuevo impulso a la programación orientada a objetos

Siguiendo con el ejemplo podríamos decir, además, que cada clase definirá las propiedades y comportamientos de sus objetos. Por ejemplo la clase *Empleados* puede tener unas determinadas propiedades como podrían ser los datos personales, la categoría, el sueldo... y unos determinados comportamientos o acciones, por ejemplo el cálculo del sueldo.

En un lenguaje más propiamente informático diríamos que una clase define las propiedades (o atributos) y métodos (los comportamientos o acciones) de todos los objetos que han sido creados (o instanciados) de la clase. De esta forma, cualquier objeto creado a partir de una clase se denominará instancia.

También podríamos decir que *Visual Basic* es un lenguaje de programación que desde siempre ha usado clases. Pensemos por ejemplo en las etiquetas o *labels*, ya que en realidad cuando estamos añadiendo un *label* a un *form* determinado lo que estamos haciendo es creando una instancia de la clase *label*.

OBTENIENDO INFORMACIÓN DE LAS CLASES

A veces necesitaremos saber a qué clase pertenece un determinado objeto para saber, por ejemplo, si le podemos aplicar una propiedad determinada o no. Para saberlo podemos utilizar las instrucciones *TypeOf* y *TypeName*. *TypeOf* sólo puede utilizarse en instrucciones del tipo *If ... Then ... Else* y debe incluir el nombre de la clase directamente en el código, como se aprecia en el siguiente ejemplo:

```
If TypeOf Command3D1 Is CheckBox
    Then ...
```

En cambio la función *TypeName* puede ser utilizada en cualquier lugar del código y devolverá el nombre de clase en formato de cadena, con lo que podremos compararla con una variable de tipo *string*.

¿CÓMO CREAR CLASES?

Ahora que tenemos una visión global sobre los objetos y cómo tratarlos vamos a ver la forma de crear las clases, para añadirlas a nuestros proyectos. Para definir o crear una clase tenemos que seguir los siguientes pasos:

1. Insertaremos un modulo de clase.
2. Definiremos sus propiedades.
3. Crearemos los métodos para la clase.
4. Crearemos los eventos.

Para insertar un módulo de clase crearemos un nuevo proyecto en *VB* y luego con el menú **Insertar**, añadiremos un *Módulo de clase*. Una vez hecho esto, editaremos las propiedades de la clase (**menú Ver** y **Propiedades** o **F4**) y asignaremos un nombre a la clase. Para nuestro ejemplo usaremos **Cprueba**.

DEFINIENDO LAS PROPIEDADES

Un módulo de clase, asimismo, podrá, como cualquier otro módulo, tener variables o elementos de datos asociados (tal y como un módulo tendría sus variables a nivel de módulo).

A cada variable o elemento le llamaremos datos miembros. Algunos de estos datos serán para guardar las propiedades de la clase y estarán accesibles desde cualquier punto del programa y otros datos miembros serán invisibles para otras partes de la aplicación ya que sólo los usará la clase dada. Lograremos este último punto declarando a dichos datos privados, con la palabra reservada *Private*.

Visual Basic es un lenguaje de programación que desde siempre ha usado clases

Además cabe decir que cada objeto creado o instanciado desde una clase tendrá una copia separada de sus datos miembros. Esto quiere decir que si cambiamos el valor de un dato de un objeto, dicho cambio no repercutirá en los otros objetos.

PROPIEDADES DE UNA CLASE

Cuando deseemos que una determinada propiedad de una clase sea visible a toda la aplicación lo lograremos con los procedimientos de propiedades. Dichos procedimientos son iguales que cualquier procedimiento de tipo *Subrutina* o *Función* pero con la característica de que están diseñados expresamente para la función de enseñar las propiedades al resto de la aplicación.

Existen principalmente tres tipos de procedimientos de propiedades:

- Procedimientos de propiedades *Get*
- Procedimientos de propiedades *Let*
- Procedimientos de propiedades *Set*

- Procedimientos de propiedades *Get*: Dicho procedimiento servirá siempre que queramos que la aplicación pueda leer los valores de una propiedad. Cabe decir que también podríamos incluir en dicho procedimiento los cálculos que quisiéramos para devolver un valor concreto.

- Procedimientos de propiedades *Let*: Este procedimiento permitirá a otras partes de la aplicación establecer un valor de una propiedad de la clase. En dicho procedimiento normalmente se suele almacenar el código responsable de la validación o conformación de datos.

- Procedimientos de propiedades *Set*: Constituye un caso especial y particular del procedimiento de propiedades *Let* y se usa cuando la propiedad de la que establecemos un valor, hace referencia a un objeto.

Cabe destacar que podemos hacer que una propiedad sea sólo de lectura o sólo de escritura modificando sus procedimientos de propiedades *Get*, *Let* o *Set*.

Listado

```
Public Property Get Categoria() As Integer
Categoria = m_Categoria
End Property

Public Property Let Categoria(Cat As Integer)
If Cat > 1 Or Cat < 10 Then
  m_Categoria = Cat
End If
End Property

Public Property Get Nombre() As String
Nombre = m_Nombre
End Property

Public Property Let Nombre(Nombre As String)
m_Nombre = Nombre
End Property

Public Property Let Nombre(Nombre As String)
m_Nombre = Nombre
End Property

Public Property Get Sueldo() As Integer
Sueldo = m_Sueldo
End Property

Public Property Let Sueldo(Sueld As Integer)
'prevenimos que el sueldo sea >0
If Sueld > 0 Then
  mSueldo = Sueld
End If
End Property
```

Vamos a ver un ejemplo de cómo implementar las propiedades. Imaginemos que queremos una clase que proporcione información de los empleados de una empresa. La información que deseamos guardar es por una parte el nombre del empleado, su sueldo y su categoría. Vamos a crear en primer lugar los datos miembros de la clase correspondientes a la información que deseamos guardar:

```
Option Explicit

Private m_Nombre As String
Private m_Sueldo As Integer
Private m_Categoria As Integer
```

Además, cabe destacar que dichos datos miembros sólo serán accesibles desde dentro de la clase, ya que son datos privados (tal y como indica la palabra reservada *Private*).

Una vez realizado dicho paso pasaremos a establecer los procedimientos de propiedades para cada propiedad o atributo de la clase, teniendo en cuenta que si queremos poner alguna condición en la adquisición de datos, tendremos que poner el código adecuado para

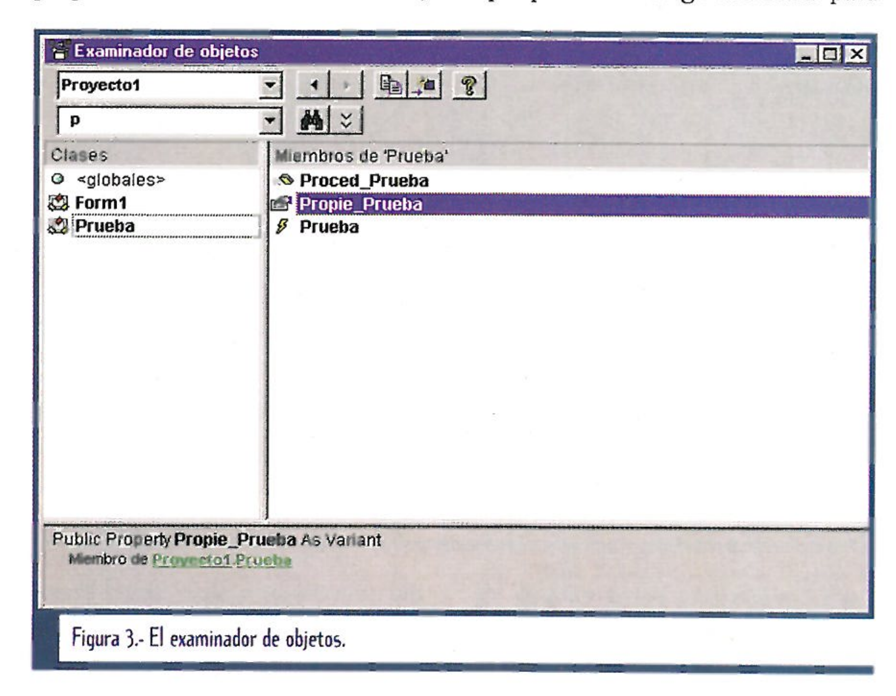


Figura 3.- El examinador de objetos.

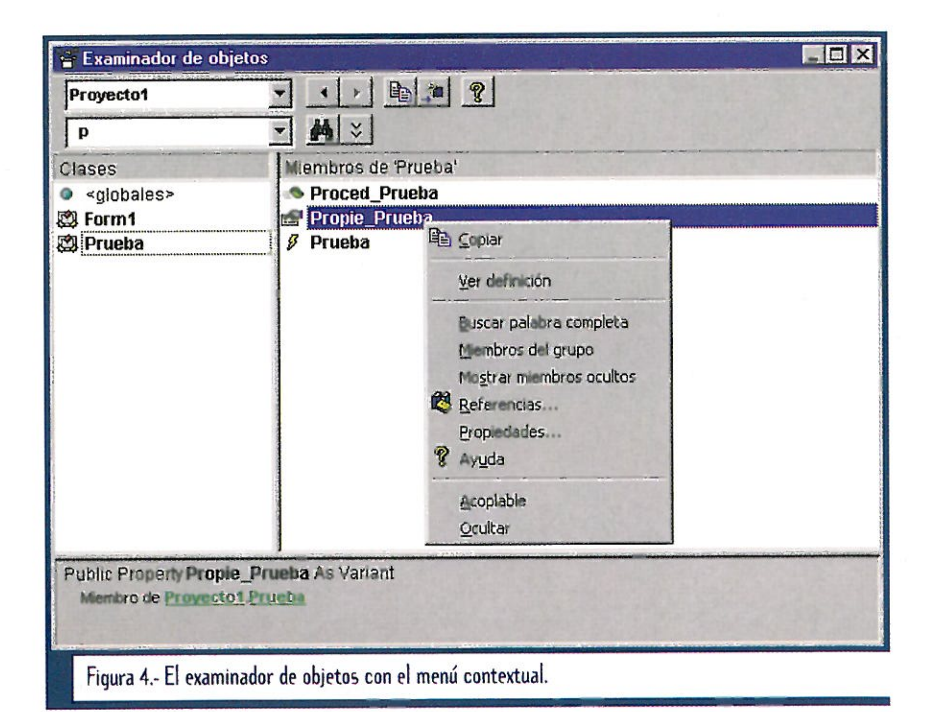


Figura 4.- El examinador de objetos con el menú contextual.

cada ocasión (ver como ejemplo el procedimiento *Let* de la propiedad *Categoria*).

Puede ser que al principio parezca un poco confuso y que sea mejor utilizar variables globales o una estructura de datos, pero debemos tener en cuenta que utilizando los procedimientos de propiedades tenemos la ventaja de que el código para la validación de datos o formato de los mismos puede ser encapsulado dentro de los procedimientos de propiedades.

Además, podemos establecer fácilmente propiedades de sólo lectura o sólo escritura (lo hacemos sin asociar el procedimiento *Get* o el *Let*). Otra ventaja sería el hecho de que podemos modificar los procedimientos de propiedades sin tener que modificar los códigos que utilizan dichas propiedades.

CREANDO LOS MÉTODOS

Los métodos de una clase definirán el comportamiento de todos los objetos creados de una clase dada. La aplicación no verá la implementación del método, aunque su funcionalidad estará visible por toda la aplicación. Es como cuando utilizamos el método *Move* del objeto *Label* comentado anteriormente, ya que aquí desconocemos cómo se ha implementado y sólo nos preocupamos de la funcionalidad que nos ofrece.

Antes de crear un método hay que determinar primero si éste va a ser público o privado

Para crear un método primero tenemos que determinar si éste va a ser público (estará accesible desde cualquier parte de la aplicación) o privado (sólo desde dentro de la clase). En segundo lugar tendremos que saber si devolverá un valor o no para definirlo como una *Subrutina* o como una función. Por último sólo nos queda escribirlo. Por ejemplo:

```
Public Sub
    Cambio_Categoria(Incremento
    As Integer)
  Categoria = Categoria +
    Incremento
End Sub
```

CREANDO LOS EVENTOS

Los módulos de clases tienen dos eventos: *Initialize* y *Terminate*. Dichos eventos aparecerán automáticamente cuando creemos una clase.

El evento *Initialize* se usa para inicializar los objetos creados desde la clase. Cuando un objeto se crea, lo primero que hará *VB* será ejecutar el código de dicho evento, incluso antes de establecer cualquier prioridad o de ejecutar cualquier método. Dicho evento sería similar a lo que en otros lenguajes de orientación a objetos sería el constructor de la clase.

Cuando todas las referencias a un objeto han sido canceladas entonces se ejecuta el evento *Terminate*.

Por ejemplo si queremos asignar una categoría inicial haríamos:

```
Private Sub Class_Initialize()
  m_Categoria = 1
End Sub
```

UTILIZANDO LA CLASE

Hasta ahora hemos visto cómo crear clases, aunque la clase por sí sola no hará nada. Nosotros trabajamos siempre con instancias de la clase, por lo tanto necesitamos el código para crear instancias de una clase. Tenemos dos formas de crear una instancia de una clase:

```
Private m_Prueba as New Cprueba
```

O bien:

```
Private m_Prueba as Cprueba
Set m_Prueba = New CPrueba
```

En la primera forma declaramos una variable y le asignamos el objeto, y en la segunda forma la primera



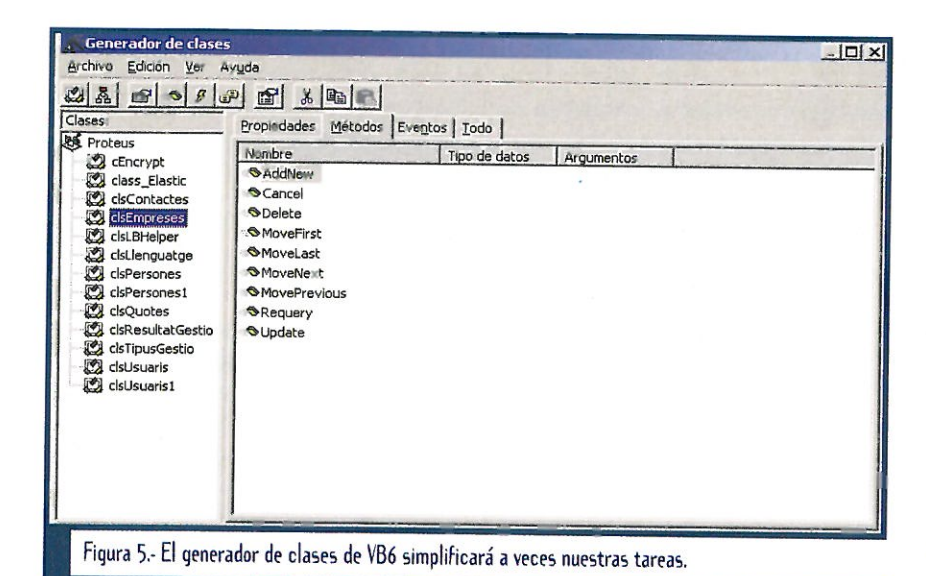


Figura 5.- El generador de clases de VB6 simplificará a veces nuestras tareas.

línea declarará la variable y la segunda asignará el objeto a la variable.

En nuestro ejemplo en particular:

```
Option Explicit

Private m_Prueba1 As CPrueba
Private m_Prueba2 As CPrueba
```

Para la declaración de variables, y el procedimiento de carga del formulario asignamos un valor:

```
Private Sub Form_Load()
  Set m_Prueba1 = New CPrueba
  Set m_Prueba2 = New CPrueba
End Sub
```

Para acceder a las propiedades de una clase el formato es el siguiente:

```
Objeto.Propiedad = Valor
```

En nuestro caso si suponemos que tenemos un botón para establecer las propiedades, entonces su código sería:

```
Private Sub Command1_Click()
  m_Prueba1.Nombre = Text
    Nombre1.Text
  m_Prueba1.Sueldo = Val
    (TextSueldo1.Text)
  m_Prueba1.Categoria = Val(Text-
    Categoria1.Text)
End Sub
```

El menú contextual es una alternativa a los botones Copiar y Ver definición del examinador de objetos

Y por último quedaría terminar la referencia al objeto que hemos creado para así liberar la memoria que hemos ocupado y los recursos del sistema. La sintaxis para realizar dicho paso sería:

```
Set Variable = Nothing
```

En nuestro ejemplo tenemos dicho código asignado a los botones **Destruir Clase 1** y **Destruir Clase 2**, pero lo más lógico cuando estamos programando es que dicho código se encuentre, por ejemplo, en el procedimiento *Form_Unload*.

DEFINIENDO COLECCIONES

Una colección de objetos es un grupo ordenado de elementos que puede ser procesado como un conjunto. Una colección puede ser referenciada desde cualquier sitio de la aplicación como cualquier otro objeto.

Para crear una colección necesitamos una clase (llamada clase de colección).

```
Option Explicit

Private m_ColPruebas As New
    Collection
```

DEPURACIÓN DE LOS MÓDULOS DE CLASE

Cuando entremos con la depuración de módulos de clase nos encontraremos con que es un poco diferente de los programas normales con los que estamos acostumbrados a trabajar. Esto sucede porque un error en un módulo de clase siempre actúa como un error controlado, que es lo mismo que decir que siempre habrá un procedimiento en la pila de llamadas que pueda controlar el error, ya sea éste el módulo de clase o el procedimiento que ha invocado la llamada a la propiedad.

Los módulos Initialize y Terminate son elementos prioritarios cuando creamos una clase cualquiera

Es decir, si una determinada función o método de una clase produce un error no se nos mostrará donde realmente se ha producido sino desde el lugar donde se ha utilizado dicha propiedad.

Para evitar esta situación *VB* ha añadido la opción de interceptar errores *Interrupción en módulos de*

clases, además de las existentes *Interrupción en errores no controlables* y *Modo de interrupción en todos*.

Dichas opciones están disponibles en el menú **Herramientas**, cuadro de diálogo *Opciones* y ficha *General*. Vamos a ver un ejemplo práctico donde el módulo de *Class1* tiene el siguiente código:

```
Public Sub Oops()
    Dim intOops as integer
    intOops = intOops  / 0
End Sub
```

Por otra parte tenemos un módulo de clase o formulario estándar que invocará al miembro *Oops*:

```
Private Sub Command1_Click()
    Dim c1 as New Class1
    c1.Oops
End Sub
```

Con el generador de clase de Visual Basic 6 simplificaremos mucho nuestro trabajo

Si activamos la opción *Interrupción en errores no controlables* la ejecución no se detendrá en la división por cero. En dicha opción el error se producirá en el procedimiento que llama *Command1_Click*. Si utilizáramos la opción *Interrupción en todos los errores* se detendría en la división por cero, pero luego nos encontraríamos con que también se para en todos los errores, incluso en los que tenemos un código de control de errores, con lo que la depuración se dificultaría.

Por eliminación obtenemos que la opción más lógica sería la de *Interrupción en módulos de clases*. Con dicha opción, el flujo del programa sólo se detendrá en los errores no controlados del módulo de clase. Si existe un controlador de errores, la ejecución no se detendrá.

Por otra parte es la opción predeterminada de *VB*. Y como nota final debemos añadir que si no existen módulos de clase dicha opción es equivalente a *Interrupción en errores no controlables*.

EL EXAMINADOR DE OBJETOS

Para ver el *Examinador de objetos*, en el menú **Ver** elegiremos la opción del mismo nombre o bien presionaremos **F2**. Dicho examinador sirve para ver la información sobre las clases, métodos, propiedades, etc.

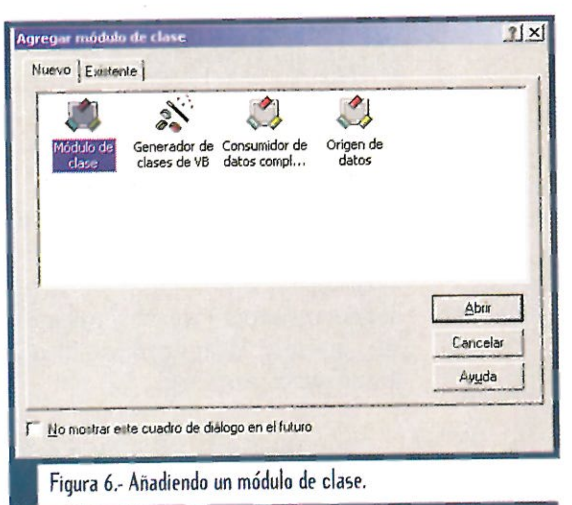


Figura 6.- Añadiendo un módulo de clase.

La información se presenta en tres niveles. Empezando por la parte superior, podremos observar los proyectos y bibliotecas disponibles, incluidos nuestros propios proyectos.

Haciendo *clic* en una clase de la lista clases veremos su descripción en el panel inferior, las propiedades, métodos, eventos y constantes aparecerán en la lista miembros de la derecha.

Si hacemos *clic* en un miembro de la lista miembros veremos los argumentos y los valores de retorno del mismo.

A través del menú contextual también podemos trabajar con el examinador. Dicho menú presenta una alternativa a los botones **Copiar** y **Ver definición del examinador de objetos.** También permite abrir el cuadro de diálogo referencias y ver las propiedades del elemento que seleccionemos. También podremos establecer descripciones de nuestros propios objetos con **Agregar descripciones de objetos**.

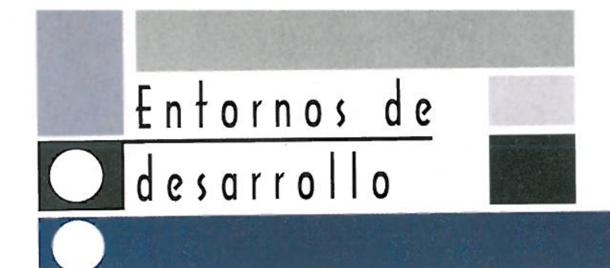
Cuando la opción *Miembros del grupo* está activada todas las propiedades de un objeto aparecen agrupadas (también los métodos, eventos, etc.) y cuando no está activada la lista de miembros es alfabética.

Necesitamos la clase de colección a la hora de crear una de ellas

Mientras que cuando aparece la opción **Mostrar miembros ocultos,** las listas clases y miembros muestran la información marcada como oculta en la biblioteca de tipos. Se muestran en color gris claro.

CONCLUSIÓN

Hasta aquí hemos visto algunas de las características básicas de las clases actuales de *VB6*. En el próximo artículo ahondaremos aún más en las características antes citadas, así como algunas exclusivas de esta nueva versión.





Visual C++ y MFC (IV)

Constantino Sánchez Ballesteros.
Técnico Superior Dearrollo Aplicaciones Informáticas.

Los menús son una parte muy importante para cualquier interfaz que utilice las ventanas como marco de operaciones. Por ello, en esta entrega profundizaremos en la creación y utilización de los menús en nuestros programas.

MENÚS

La mayoría de las aplicaciones creadas para sistemas operativos bajo entorno gráfico utilizan menús para dotar de funcionalidad al programa (sin enumerar los diversos controles que se inserten en el formulario).

Si un usuario pulsa sobre el título de un menú, automáticamente se despliega mostrando las opciones que contiene. Estas opciones pueden ser órdenes o submenús. Por supuesto, cada submenú puede tener a su vez más opciones que pueden desplegarse.

CREACIÓN DE MENÚS

Para crear un menú podemos utilizar el *Editor de Menús* de *Visual Studio*. Los pasos de la creación son los siguientes:

- Abrir el **Editor de Menús**: una vez dentro de nuestro proyecto, seleccionaremos la pestaña **ResourceView** de la ventana *WorkSpace* y pulsaremos sobre la carpeta **Menú**. En ella aparecerá el recurso *IDR_MAINFRAME*, que es el recurso que utiliza la aplicación para la utilización de un menú. Pulsando sobre este recurso aparecerá automáticamente el menú por defecto que creó para nosotros el asistente de *Visual Studio*.

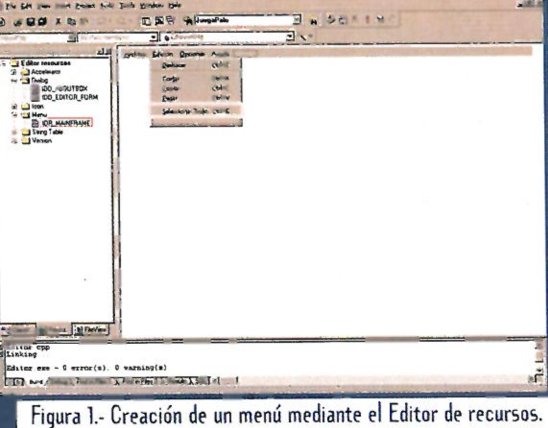
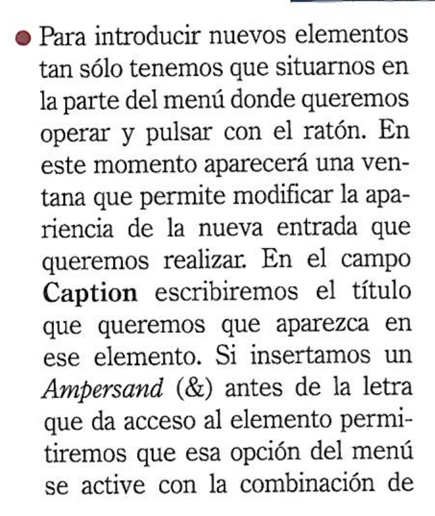


Figura 1.- Creación de un menú mediante el Editor de recursos.



- Para introducir nuevos elementos tan sólo tenemos que situarnos en la parte del menú donde queremos operar y pulsar con el ratón. En este momento aparecerá una ventana que permite modificar la apariencia de la nueva entrada que queremos realizar. En el campo **Caption** escribiremos el título que queremos que aparezca en ese elemento. Si insertamos un *Ampersand* (&) antes de la letra que da acceso al elemento permitiremos que esa opción del menú se active con la combinación de teclas compuesta por la tecla **Alt** y la letra en cuestión. El campo *ID* se utiliza para que al pulsar sobre esa opción del menú se ejecute alguna acción predeterminada (*Guardar*, *Abrir*, etc.). Por ejemplo, *ID_APP_EXIT* es el identificador de la orden **Salir** del menú **Archivo**, y ya está implementada de forma automática.

- Creación de submenús: un elemento de un menú puede ser una orden o un submenú. Para que un elemento se comporte como un



submenú debemos asignar a **True** la propiedad *pop-up* del mismo.

- Separadores: podemos utilizarlos para agrupar las órdenes en función de sus acciones. Para insertarlo debemos seleccionar un rectángulo vacío del menú y activar la propiedad **Separator** en el cuadro de diálogo de **propiedades de menú.**

- Guardar el menú: una vez creado el menú a nuestro antojo ejecutaremos la orden **Save** del menú **File** de *Visual Studio* y cerraremos la ventana correspondiente del menú. Con esto habremos guardado el menú en una plantilla de recursos de nuestro proyecto que utiliza nuestro entorno de programación.

Para eliminar entradas de menú utilizaremos la tecla Supr

Si en algún momento queremos borrar un menú o algún elemento del mismo, lo seleccionaremos y pulsaremos la tecla **Supr** o **Del** (borrar).

En principio, los elementos que componen un menú no estarán operativos hasta que se una el código correspondiente. De todos modos, los identificadores predefinidos (comentados anteriormente) tienen operatividad desde el primer momento ya que el asistente escribe el código necesario automáticamente.

Para que una orden del menú responda a una pulsación de botón deberemos escribir una función para ella. Las órdenes de un menú sólo responden al evento *clic* del ratón o a la tecla **Intro.** Si el usuario selecciona una orden, se enviará un mensaje *WM_COMMAND* a nuestra aplicación.

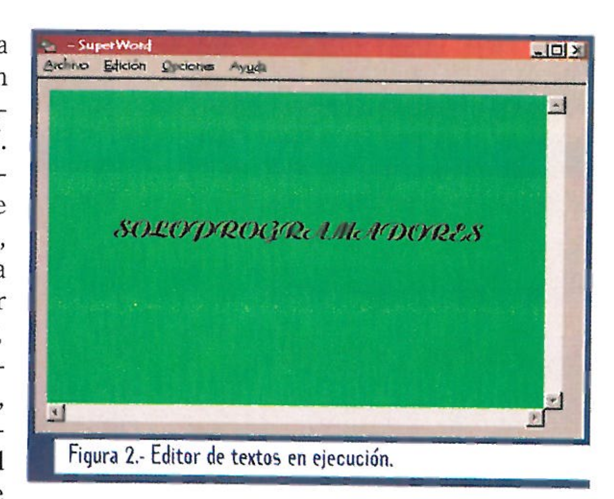


Figura 2.- Editor de textos en ejecución.

Para escribir una función para una orden de nuestro menú utilizaremos *ClassWizard.* Una vez allí seleccionaremos la clase que manipulará la orden, normalmente nuestra vista, el identificador del menú *(ID_xx_xx,* donde *xx_xx* es la acción que ejecutar), *COMMAND* y pulsaremos sobre el botón **Add Function.** Con esto se escribirá automáticamente en el archivo de nuestra vista el código de la función correspondiente. El siguiente ejemplo muestra la creación automática de código para la orden *ID_FILE_NEW*:

```
void CEditorView::OnFileNew()
{
   // TODO: Add your command
      handler code here
}
```

PROPIEDADES

A continuación se muestran las propiedades que podemos seleccionar en la ventana de diseño de menús:

- *ID*: Permite la identificación de un menú o uno de sus elementos en nuestro código.

- *Separator*: Es una línea de separación entre dos elementos de un menú.

- *Checked*: Indica el estado de la opción a la que hace referencia la orden. Si vale **True**, aparecerá una marca √ a la izquierda del elemento del menú.

- *Pop-up*: permite que un elemento se convierta en un submenú.

- *Grayed*: se utiliza para activar/desactivar una orden en un momento determinado. El texto de la orden aparecerá en tono gris.

- *Inactive*: Funciona igual que *Grayed* pero el texto de la orden aparece normal.

- *Help*: Permite que un menú aparezca en el margen derecho (útil para los menús tipo *Ayuda*).

- *Break*: Toma varios valores para establecer el formato del menú.

- *Prompt*: Contiene el texto que aparece en la barra de estado cuando el elemento es seleccionado.

PROGRAMA DE EJEMPLO

La aplicación que vamos a realizar para ver la funcionalidad de los menús será un sencillo procesador de textos. Aunque en principio pueda parecer complicado, un sencillo procesador de textos es fácil de hacer bajo *MFC's*.

Mediante una caja de texto multilínea lograremos dar funcionalidad a nuestro editor. Estas cajas de texto dependen en gran medida de varias propiedades, entre ellas *Multiline, horizontal Scroll, vertical Scroll* y *Want return.* Estas propiedades, cuando se crean mediante una plantilla, sólo pueden editarse en la fase de diseño del programa.

Multiline permite utilizar cajas de texto con múltiples líneas. Si deseamos que se inserte en el texto un retorno de carro más avance de línea, tenemos que establecer la propiedad *Want return*.

Al crear nuevas órdenes de menú, debemos editar sus propiedades

Horizontal scroll proporciona una barra de desplazamiento horizontal y *vertical scroll* insertará una barra vertical. Si en una caja de texto multilínea no ponemos una barra horizontal y la propiedad *Auto Hscroll* no está activada, la línea de texto que alcance el extremo derecho de la caja continuará de forma automática en la siguiente línea.

A continuación se detallan los pasos que seguir para construir nuestra aplicación:

- Mediante el asistente, creamos un nuevo proyecto para una aplicación *SDI* con un formulario principal.
- Con el **Editor de diálogos** insertaremos en el formulario una caja de texto del tamaño que creamos conveniente. Recordemos que este control será el que utilicemos para escribir el texto.
- *Multiline* debemos establecer con el valor **True**. Con esto lograremos escribir en la caja varias líneas de texto.
- *Horizontal* y *Vertical Scroll* también tendrán que establecerse a **True**. Con esto lograremos que nuestra caja de texto tenga barras de desplazamiento horizontal y vertical.
- *Want return* tenemos que ponerla a **True** para que salte a una línea nueva cuando se pulse la tecla **Intro**.
- Poniendo *Border* a **False** eliminaremos el borde de la caja de texto (opcional).

Es el momento de salvar la aplicación y ejecutarla. Como podremos comprobar, sin escribir nada de código ya tenemos la funcionalidad básica que permite cualquier editor de textos sencillo tipo *Notepad* de *Windows*.

El siguiente paso será añadir las órdenes *copiar, pegar, cortar* y *deshacer* al menú de la aplicación. Para ello utilizaremos las siguientes funciones:

- *Undo*: Permite deshacer la última operación realizada sobre el control de edición.
- *Clear*: Borra el texto seleccionado.
- *Copy*: Copia el texto seleccionado.
- *Cut*: Corta el texto seleccionado.
- *Paste*: Inserta el texto que contenga el portapapeles, a partir del punto de inserción en el que estemos.
- *GetSel*: Obtiene las posiciones del primer caracter seleccionado y del primero no seleccionado.
- *SetSel*: Permite seleccionar un rango de caracteres en el control.

Para poder copiar y pegar texto en el control tendremos que utilizar el portapapeles de *Windows*. Éste contiene la última copia de la información que le hayamos enviado. Los datos contenidos en el portapapeles suelen tener un formato particular (texto, *bitmaps*, etc.). Algunos de estos formatos se muestran a continuación:

- *CF_TEXT*: para cadenas de caracteres *ANSI* terminadas en **0**.
- *CF_BITMAP*: para mapas de *bits* (imágenes).
- *CF_METAFILEPICT*: para imágenes de meta-archivo. Un meta-archivo guarda los datos de la imagen como una serie de registros que se corresponden con llamadas al *GDI* (*Rectangle*, *DrawText*, etc.).

Retomando el hilo de nuestra aplicación, es el momento de crear el menú con nuestros requisitos. Tenemos que crear las siguientes entradas de menú:

1. Archivo
2. Edición
3. Ayuda

En el menú **Archivo** pondremos la orden **Salir**. En el de **Edición** insertaremos las órdenes **Deshacer**, **Cortar**, **Copiar** y **Pegar**. Para finalizar, en el menú ayuda pondremos la orden **Acerca de.**

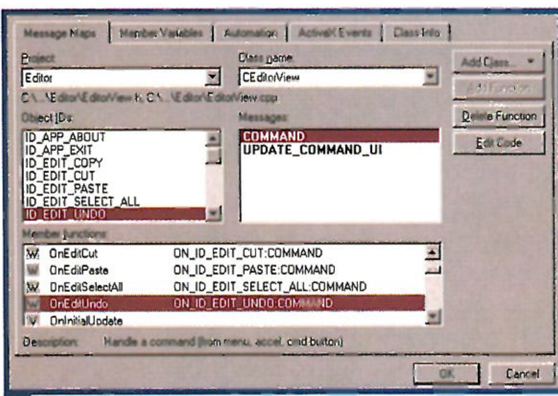


Figura 3.- Creación de funciones para identificadores mediante *Classwzard*.



TABLA 1. Ejemplo de pulsación de teclas.

OBJETO	PROPIEDAD	VALOR
Archivo	Caption	&Archivo
	Pop-up	True
Salir	ID	ID_APP_EXIT
	Caption	&Salir
Edición	Caption	&Edición
	Pop-up	True
Deshacer	ID	ID_EDIT_UNDO
	Caption	&Deshacer
	Grayed	True
Cortar	ID	ID_EDIT_CUT
	Caption	Cor&tar
	Grayed	True
Copiar	ID	ID_EDIT_COPY
	Caption	&Copiar
	Grayed	True
Pegar	ID	ID_EDIT_PASTE
	Caption	&Pegar
	Grayed	True
Ayuda	Caption	&Ayuda
	Pop-up	True
	Help	True
Acerca de	ID	IDD_APP_ABOUT
	Caption	&Acerca del editor de textos

En la Tabla 1 se muestran las propiedades que deben tener los menús y sus correspondientes órdenes.

Para poder añadir una orden del menú a nuestro código debemos utilizar el asistente de clases (*ClassWizard*). Primero debemos seleccionar la orden del menú con la que queremos operar y seguidamente pulsar **Ctrl+W**.

Comenzaremos por la orden *Deshacer*. Como ya hemos comentado en alguna ocasión, *Windows* envía mensajes a nuestra aplicación siempre que ocurra algún evento. Si el usuario pulsa sobre la orden del menú deshacer, *Windows* enviará los mensajes *WM_MENUSELECT* y *WM_COMMAND*. Para poder manipular *WM_COMMAND* debemos seleccionar en la lista *Object IDs* el objeto *ID_EDIT_UNDO* y en la lista *Messages* el mensaje *COMMAND*. Seguidamente pulsaremos en el botón **Add Function** y añadiremos la función miembro *OnEditDeshacer*.

Una vez realizados estos pasos se creará de forma automática el siguiente código:

```
void CeditorView::OnEditDeshacer()
    {
// TODO. ADD YOUR CODE HERE
}
```

ACTIVACIÓN DE MENÚS

Como podemos ver en muchos programas, cuando se activa un

menú, hay muchas opciones que aparecen activadas y otras desactivadas. Este hecho lo debemos controlar nosotros en nuestra aplicación.

Cuando se despliega un menú, *Windows* envía los mensajes de iniciación *WM_INITMENU* y *WM_INITMENUPOPUP* que en las *MFC's* se manipulan a través de la macro *ON_UPDATE_COMMAND_UI*. Con el uso de esta macro sabremos cuándo debemos actualizar un menú.

Cualquier editor de textos debe tener al menos las opciones de cortar, copiar, pegar y deshacer

Para saber si la orden *Deshacer* (anteriormente creada) debe activarse, tendremos que invocar a la función miembro *CanUndo* de la clase *CEdit*. *CanUndo* devuelve un valor distinto de cero si el control de edición puede realizar la acción de deshacer.

Lo primero que tenemos que hacer es abrir *ClassWizard* y añadir la función *OnUpdateEditDeshacer* que está vinculada al objeto *ID_EDIT_UNDO*, para poder manipular el mensaje *UPDATE_COMMAND_UI*. Seguidamente escribiremos el código expuesto a continuación. Previamente tenemos que vincular el control *IDC_EDIT1* con una variable del tipo *CEdit* con el nombre que deseemos (para este ejemplo se utiliza *m_Edit1*).

```
void CeditorView::OnUpdateEdit
    Deshacer(CcmdUI *pCmdUI)
{
    pCmdUI-
    >Enable(m_Edit1.CanUndo());
}
```

TABLA 2. Formatos soportados por el portapapeles.

Valor	Descripción
CF_BITMAP	*Handle* a un *bitmap (HBITMAP)*.
CF_DIB	Objeto que contiene la estructura *BITMAPINFO* seguida por los *bits* del *bitmap*.
CF_LOCALE	El dato es un *handle* a un identificador local asociado con el texto del portapapeles.
CF_METAFILEPICT	*Handle* de un gráfico de tipo *metafile*.
CF_OEMTEXT	Formato de texto conteniendo caracteres en formato *OEM*. Cada línea finaliza con un retorno de carro y un salto de línea.
CF_PALETTE	*Handle* a una paleta de color. Cuando se visualizan datos del portapapeles, éste siempre utiliza como paleta actual cualquier objeto del portapapeles que esté en el formato *CF_PALETTE*.
CF_PRIVATEFIRST THROUGH CF_PRIVATELAST	Rango de valores enteros para formatos de portapapeles privados.
CF_RIFF	Representa datos de audio más complejos que los que pueden representarse a través del formato estándar *CF_WAVE*.
CF_SYLK	Formato para *Microsoft Symbolic Link (SYLK)*.
CF_TEXT	Formato de texto. Cada línea finaliza con un retorno de carro y un salto de línea.
CF_WAVE	Representa datos de audio en uno de los formatos de onda estándar, como pueden ser 11 *Khz* o 22 *Khz (PCM)*.
CF_TIFF	Formato de imagen *TIFF*.
CF_UNICODETEXT	**Windows NT:** formato de texto *Unicode*. Un carácter nulo significa el final de los datos.

Debemos tener en cuenta que debemos utilizar un manejador *ON_UPDATE_COMMAND_UI* por cada orden de menú. Cuando se despliega un menú, el sistema busca y llama a cada manejador, el cual llama a las funciones miembro de *CcmdUI* (*Enable*, *SetText*, etc.) y visualiza adecuadamente cada elemento del menú.

CcmdUI sólo se utiliza en las funciones miembro manipuladoras de mensajes relacionados con la macro *ON_UPDATE_COMAND_UI* y derivadas de clases *Ccmd Target*.

La función *Enable* permite activar o desactivar un elemento de un menú, un botón de la barra de herramientas, etc. Retomando nuestra aplicación, para activar las órdenes **Cortar** y **Copiar** se necesita que el usuario haya seleccionado un bloque de texto. Esto podemos averiguarlo utilizando la función *GetSel* de la clase *CEdit*. El formato de esta función es el siguiente:

```
void GetSel(int &nPosInicial, int
    &nPosFinal) const;
```

- *nPosInicial*: es la posición del primer caracter que hemos seleccionado.
- *nPosFinal*: es la posición del primer caracter no seleccionado.

Siguiendo los pasos anteriores para la orden del menú *Deshacer*, tendremos que añadir las siguientes funciones miembro:

```
void CeditorView::OnUpdateEdit
    Cortar(CcmdUI *pCCmdUI)
{
   int nPosInicial, nPosFinal;
   m_Edit1.GetSel
     (nPosInicial,nPosFinal);
   pCmdUI->Enable ( (nPos
```

```
    Final-nPosInicial) >0 );
}

void CeditorView::OnUpdateEdit
    Copiar(CcmdUI *pCCmdUI)
{
   int nPosInicial, nPosFinal;
   m_Edit1.GetSel
    (nPosInicial,nPosFinal);
   pCmdUI->Enable ( (nPos
    Final-nPosInicial) >0 );
}
```

La orden *Pegar* se activará cuando haya texto en el portapapeles y se desactivará cuando se encuentre vacío. Para poder determinarlo utilizaremos la función *IsClipboard FormatAvailable*.

```
void CeditorView::OnUpdateEdit
    Pegar(CcmdUI *pCCmdUI)
{

    IsClipboardFormatAvailable(CF
    _TEXT)
     ? pCmUI->Enable(True)
     : pCmUI->Enable(false);
}
```

En la Tabla 2 podemos ver al detalle los formatos más importantes soportados por la función *IsClipboardFormatAvailable.*

Es el momento de procesar las órdenes de los menús. Comenzaremos por *Deshacer*. Para poder deshacer una operación efectuada en una caja de texto debemos utilizar la función *Undo* de la clase *CEdit.* Anteriormente creamos mediante *ClassWizard* la función *OnEdit Deshacer*, la cual utilizaremos para gestionar la función *Undo.* Por lo tanto iremos al código y editaremos:

```
void CeditorView::OnEditDeshacer()
{
    m_Edit1.Undo();
}
```

El siguiente paso lo enfocaremos a la orden *Cortar*. Al igual que para *Deshacer,* debemos utilizar *ClassWizard* y añadir la función *OnEditCortar* asociada con el objeto *ID_EDIT_CUT* para manipular el mensaje *COMMAND*. El código debe quedar como sigue:

```
void CeditorView::OnEditCortar()
{
    m_Edit1.Cut();
}
```

Debemos realizar los mismos pasos para las órdenes *Copiar* y *Pegar.* A continuación se muestra cómo quedaría el código:

```
void CeditorView::OnEditCopiar()
{
    m_Edit1.Copy();
}

void CeditorView::OnEditPegar()
{
    m_Edit1.Paste();
}
```

En este momento podemos ejecutar la aplicación y comprobar la funcionalidad realizada hasta ahora.

Para finalizar la aplicación sólo queda desarrollar el menú de ayuda. Como hemos visto en la fase de diseño, este menú se compone de una sola orden (Acerca del Editor de textos). Cuando creamos mediante el asistente el esqueleto de la aplicación se creó automáticamente una caja de diálogo para la visualización del *Copyright,* nombre del programa, etc. Nuestra labor consistirá en modificarla acorde con nuestras necesidades.

Lo primero que vamos a hacer será asociar un icono a la aplicación. Para ello, abrimos el **Editor de diálogos** y realizamos los siguientes pasos:

1. En la ventana **Workspace** debemos seleccionar el recurso **Icon**. Aparecerá el recurso *IDR_MAINFRAME*. Este último tendremos que eliminarlo con la tecla **Supr**.

2. Volvemos a seleccionar el tipo de recurso **Icon** y pulsamos el botón derecho del ratón. Dentro del menú contextual que aparece seleccionaremos la orden **Import**. En este punto tendremos que navegar por el disco duro para seleccionar el icono (archivo con extensión *ICO*) que creamos conveniente. Con esto lograremos que se visualice el nuevo icono.

3. Para finalizar, en la ventana **Workspace** seleccionaremos el nuevo recurso importado y abriremos sus propiedades mediante el botón derecho del ratón. Dentro de la lista *ID* seleccionaremos el identificador *IDR_MAINFRAME* y tendremos el trabajo terminado.

En la aplicación de ejemplo que acompaña la revista se han efectuado algunas implementaciones más para que las estudie el lector (cambio de fuente de letra y tamaño, etc.).

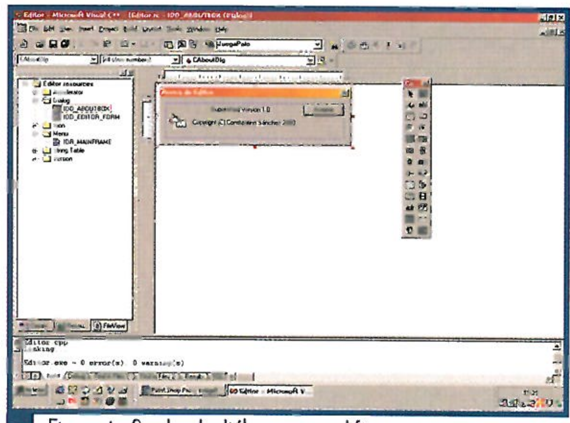
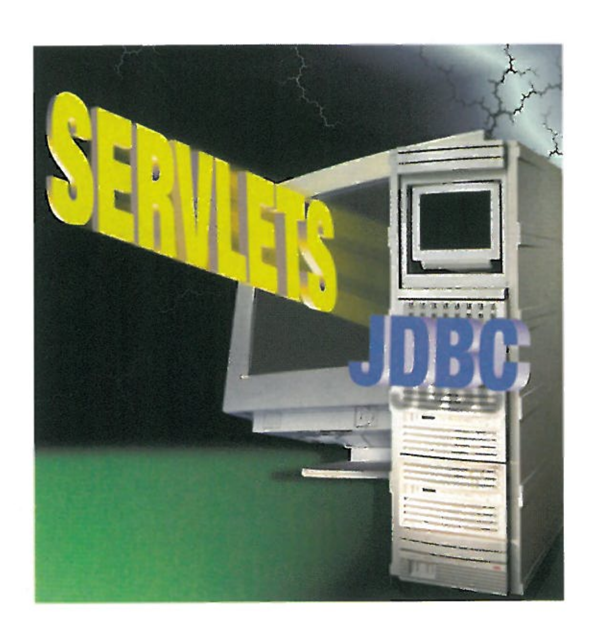


Figura 4.- Cuadro de diálogo que modificaremos para personalizar la aplicación.



Aplicaciones Web con acceso a BBDD basadas en Java (V)

Adolfo Aladro García. *Analista Programador*

Aparte de los aspectos básicos de la *API JDBC*, existen otras muchas características que conviene conocer para que nuestras aplicaciones *Web* ofrezcan el mayor rendimiento posible a la hora de acceder a fuentes de datos.

SENTENCIAS SQL PREPARADAS

En capítulos anteriores hemos trabajado con la interfaz *Statement* para crear consultas *SQL* que se lanzan contra la base de datos. Pero la *API JDBC* proporciona otra interfaz llamada *PreparedStatement* con la que podemos trabajar con sentencias *SQL* preparadas y con parámetros.

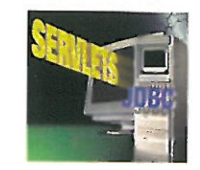
Cada vez que se ejecuta una sentencia *SQL* el gestor de la base de datos sigue una serie de pasos para establecer la estrategia de acceso. De esta forma se optimiza el rendimiento de las consultas. Cuando las sentencias *SQL* son iguales o similares, el proceso que realiza el gestor da lugar a la misma estrategia de acceso u optimización. Esto significa que el procesamiento de las consultas será notablemente más rápido si se calcula la estrategia de acceso una sola vez durante la primera sentencia, en vez de forzar al gestor de la base de datos a hacerlo por cada una de las consultas.

Este es el objeto de la interfaz *PreparedStatement.* La primera vez que se ejecuta una sentencia *SQL* declarada como *PreparedStatement* se indica al gestor de la base de datos que deseamos que optimice el acceso, y que el resto de las veces utilice la misma información en la estrategia de acceso.

Otra de las características más interesantes de la interfaz *PreparedStatement* es pasar parámetros a la sentencia *SQL*, por ejemplo para buscar en una base de datos por identificadores, especificando cada vez un identificador distinto.

Para crear una sentencia *SQL* del tipo *PreparedStatement* debemos utilizar el método *prepareStatement* de la interfaz *Connection:*

```
public abstract PreparedStatement
     prepareStatement(String sql)
          throws SQLException
```



Este método crea una sentencia preparada. El parámetro que recibe se corresponde con la sentencia *SQL* que ejecutar. Dentro de esta sentencia puede aparecer el carácter cierre de interrogación (?) en el lugar donde van a aparecer el/los parámetro/s.

LA INTERFAZ PREPAREDSTATEMENT

La interfaz *PreparedStatement* guarda muchas similitudes con la interfaz *Statement*. Sin embargo, como puede deducirse del esquema de la Figura 1, la estructura general de un acceso a base de datos con esta interfaz se modifica ligeramente.

Los parámetros se escriben con los métodos setXXX

Existe una familia de métodos que sirven para especificar el valor de los parámetros de las sentencias *SQL*. El primer argumento se corresponde con la posición del parámetro dentro de la sentencia *SQL*, comenzando por 1, y el segundo es el tipo del parámetro *(String, int,...)*.

```
public abstract void
     setString(int parameterIndex,
     String x)
          throws SQLException
public abstract void setInt(int
     parameterIndex, int x)
          throws SQLException
...
```

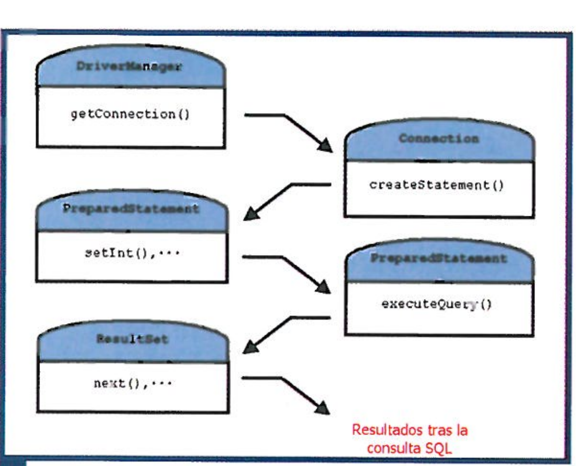


Figura 1.- Esquema de una consulta a la base de datos mediante una sentencia *SQL* preparada y con parámetros.

Las sentencias preparadas con parámetros se ejecutan mediante el método *executeQuery*. Éste ejecuta la sentencia *SQL* preparada y devuelve el resultado mediante la interfaz *ResultSet*. Nótese que aquí no se pasa como argumento una cadena con la sentencia que ejecuta, dado que ésta se especifica en la llamada al método *prepareStatement*, y no es modificable.

```
public abstract ResultSet execu-
     teQuery()
          throws SQLException
```

Por último, el método *executeUpdate* ejecuta una sentencia *UPDATE, DELETE, INSERT* o cualquier otra sentencia *SQL* que no devuelva un conjunto de registros. El resultado que se obtiene es el número de registros afectados por la sentencia, o **–1** si no los hubo.

```
public abstract int execute
     Update()
          throws SQLException
```

Veamos un primer ejemplo de utilización de la interfaz *PreparedStatement*. Para ello, vamos a recurrir a la misma base de datos que utilizamos en la tercera entrega de la serie, que contenía los datos de una colección de discos.

Lo primero que haremos será crear un *array* con los nombres de los artistas que buscamos en la base de datos.

```
String[] artistas = {"Madonna",
     "Beck","Moby"};
```

La sentencia preparada con parámetros que vamos a lanzar contra la base de datos será:

```
String sentencia =
     "SELECT DISCO.* FROM DISCO
     WHERE DISCO.DIS_ARTISTA= ?";
```

En el lugar donde aparece el carácter cierre de interrogación (?) se irán insertando cada uno de los elementos del *array* anterior. Crear la sentencia preparada es tan sencillo como hacer:

```
PreparedStatement select =
     conexion.prepareStatement
     (sentencia);
```

A continuación se comienza a ejecutar un *bucle* que recorre el *array* de artistas. Con cada uno de ellos utiliza el método *setString* de la interfaz *PreparedStatement* para sustituir en la posición adecuada el carácter cierre de interrogación (?) por la cadena correspondiente.

```
ResultSet resultadoSelect =
     null;for(int i=0; i
     <artistas.length; i++) {
     select.setString(1,
     artistas[i]);     ..}
```

Los pasos que siguen son idénticos a los llevados a cabo cuando utilizamos la interfaz *Statement*. Se ejecuta la sentencia:

```
resultadoSelect = select.execute
     Query();
```

y se recorre el conjunto de los resultados:

```
boolean seguir = resultado
     Select.next();while (seguir)
     {  System.out.print(resultado
     SelecgetString(2) + " ");
  System.out.print(resultado
     Select.getString(3) + " ");
```

```
System.out.println(resultado
    Select.getString(4));
  seguir =
    resultadoSelect.next();
}
resultadoSelect.close();
```

INSERCIÓN, ACTUALIZACIÓN Y ELIMINACIÓN

Como ya hemos visto, la interfaz *PreparedStatement* cuenta con el método *executeUpdate* para actualizar la base de datos, ya sea realizando una inserción, una modificación o bien borrando uno o varios registros. Aunque esto mismo puede llevarse a cabo mediante la interfaz *Statement*. Este tipo de operaciones encajan con frecuencia en el prototipo de consultas a la base de datos que conviene realizar mediante sentencias preparadas con parámetros. Estas consultas suelen aparecer en bloque, dándose varias inserciones, eliminaciones o modificaciones en cascada.

El archivo *InsertarYMostrar.java* contiene el código fuente del *Servlet* que se encarga de mostrar en la pantalla del navegador la página que ilustra la Figura 4. Ésta se encuentra divida en dos secciones: la superior sirve para que usuario pueda insertar datos en la base de datos y la inferior muestra el contenido de dicha base de datos. Si observamos el código fuente de la página mostrada por el *Servlet* podremos ver que el formulario de la sección superior se dirige al mismo *Servlet:*

```
<FORM
    ACTION="http://127.0.0.1/serv
    let/InsertarYMostrar"
      METHOD="post">
```

Esto es así porque vamos a utilizar el mismo *Servlet* para mostrar los datos y para insertarlos, en el caso de que proceda.

Lo primero que hacemos es inicializar las variables que van a recoger los parámetros en el caso de que éstos existan:

```
String DIS_ARTISTA
    = null;
String DIS_TITULO
    = null;
String DIS_FECHA =
    null;
```

Se considera que si existe al menos uno de los parámetros, por ejemplo *DIS_ARTISTA,* el usuario ha introducido datos en el formulario y ha hecho **clic** en el botón **Insertar.** Hay que realizar una inserción en la base de datos antes de pasar a mostrar el contenido de la misma.

```
DIS_ARTISTA = req.getParameter
    ("DIS_ARTISTA");
if (DIS_ARTISTA != null) {
  // Inserción en la BBDD
  ...
}
```

Para llevar a cabo la inserción en primer lugar tomamos el resto de los parámetros:

```
DIS_TITULO = req.getParameter
    ("DIS_TITULO");
DIS_FECHA =
    req.getParameter("DIS_FECHA")
    ;
```

Después construimos la cadena de texto correspondiente a la sentencia preparada, creamos dicha sentencia, le damos los parámetros y finalmente la ejecutamos:

```
String sentencia =
"INSERT INTO DISCO ( DIS_ARTISTA,
    DIS_TITULO, DIS_FECHA )
     VALUES (?, ?, ?)";
PreparedStatement select =
    conexion.prepareStatement
    (sentencia);
select.setString(1, DIS_ARTISTA);
select.setString(2, DIS_TITULO);
select.setString(3, DIS_FECHA);
int resultado = select.
    executeUpdate();
```

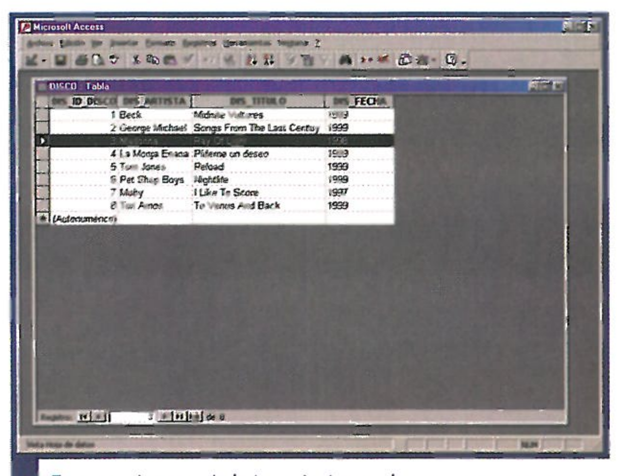


Figura 2.- Aspecto de la base de datos a la que vamos a atacar desde los *Servlets*.

En este ejemplo en concreto no es preciso utilizar la interfaz *PreparedStatement* ya que en realidad sólo se realiza una inserción. Con varias inserciones seguidas del mismo tipo, los pasos a seguir son equivalentes, y entonces sí es conveniente utilizar la interfaz *PreparedStatement.*

Las sentencias preparadas minimizan el trabajo que hemos de realizar

El resto del *Servlet* es común a las dos funciones, ya que se muestra el contenido de la base de datos tanto si se inserta como si no. En esta ocasión para realizar la consulta a la base de datos se utiliza la interfaz *Statement* tal y como sigue:

```
Statement select = conexion.
    createStatement();
ResultSet resultadoSelect =
```



```
     select.executeQuery("SELECT *
     FROM DISCO ORDER BY
     DISCO.DIS_ARTISTA");
boolean seguir = resultado
     Select.next();
...
```

Para destacar cuál es el registro que acaba de insertarse, el *bucle* principal que lee los datos devueltos por la consulta va a ser alterado. Se almacenan en variables temporales el nombre del artista, el título del disco y el año de producción en cada vuelta del bucle.

```
String dis_artista = null;
String dis_titulo = null;
String dis_fecha = null;
while (seguir) {
  ...
  dis_artista =
    resultadoSelect.getString(2);
  dis_titulo =
    resultadoSelect.getString(3);
  dis_fecha =
    resultadoSelect.getString(4);
  ...
}
```

Si el parámetro *DIS_ARTISTA* que recibió el *Servlet* es *Null* se trata de un caso en el que no ha habido inserción alguna, y por tanto no hay nada especial que reseñar en la tabla que muestra los contenidos de la base de datos. En otro caso, comparamos el disco que acabamos de extraer de la base de datos con el que se recibió desde el formulario. Si se produce la coincidencia entonces hacemos notar que ése es el registro que acaba de insertarse.

```
if (DIS_ARTISTA == null) {
  ...
} else {
  if ((DIS_ARTISTA.compareTo
    (dis_artista) != 0) ||
      (DIS_TITULO.compareTo
    (dis_titulo) != 0) ||
      (DIS_FECHA.compareTo
    (dis_fecha) != 0)) {
      ...
  } else {
      ...
  }
}
```

Para eliminar los registros, utilizaremos el fichero *InsertarYMostrar2.java* que contiene esta segunda versión. Asimismo el fichero *EliminarDiscos.java* es el *Servlet* encargado de eliminar los registros seleccionados por el usuario en la página principal. La Figura 4 muestra el nuevo aspecto de esta página.

GetParameterValues se usa para parámetros de valores múltiples

A la derecha de cada uno de los discos se sitúa una casilla de verificación. Todas las casillas de verificación se escriben de la misma forma desde el *Servlet:*

```
...
dis_id_disco =
     resultadoSelect.getString(1);
...
out.println("<TD><INPUT
     TYPE=\"checkbox\"
     NAME=\"DIS_ID_DISCO\"
     VALUE=\"" + dis_id_disco +
     "\"></TD>");
```

Nótese que la barra inclinada (\) se coloca a la izquierda de las dobles comillas cuando éstas deben emplearse mediante secuencias de *espape*.

Todas las casillas de verificación tienen el mismo nombre, si el usuario elige varias de ellas, el parámetro *DIS_ID_DISCO* que llegue al *Servlet EliminarDiscos.java* deberá ser tratado como múltiple. Esta es la razón por la que este parámetro se lee así:

```
string[] DIS_ID_DISCO =
    req.getParameterValues
     ("DIS_ID_DISCO");
```

El método *getParameterValues* devuelve un *array* de cadenas de texto, cada una de las cuales se corresponde con un valor del parámetro. En nuestro caso particular cada cadena es un identificador de un disco. La cadena que contiene la sentencia *SQL* encargada de borrar los registros es la siguiente:

```
string sentencia =
  "DELETE * FROM DISCO WHERE
    DIS_ID_DISCO = ?";
```

La sentencia preparada se crea tal y como hemos descrito antes:

```
PreparedStatement select =
     conexion.prepareStatement
     (sentencia);
```

Por último el *bucle* encargado de eliminar los registros queda como sigue:

```
for(int i=0;
     i<DIS_ID_DISCO.length; i++) {
```

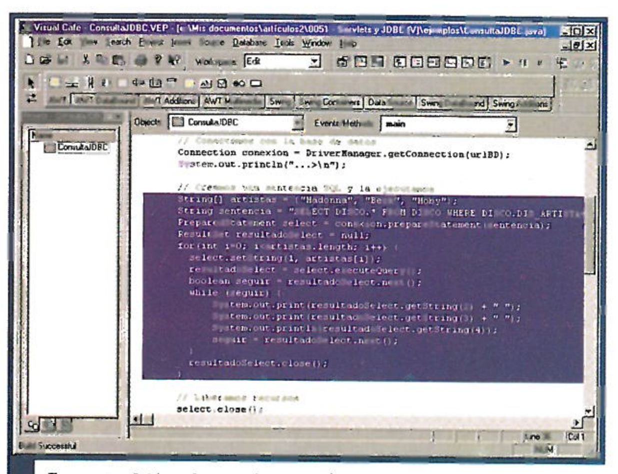


Figura 3.- Código fuente de una aplicación *Java* que utiliza sentencias preparadas para consultar la base de datos.

TABLA 1. Constantes de la interfaz Connection que determinan el nivel de aislamiento de las transacciones.

Constante	Descripción
TRANSACTION_NONE	No se pueden utilizar transacciones.
TRANSACTION_READ_COMMITTED	Desde esta transacción no se pueden ver registros que han sido modificados por otra transacción, y no guardados.
TRANSACTION_READ_UNCOMMITTED	Se ven sólo las modificaciones ya guardadas hechas por otras transacciones.
TRANSACTION_REPEATABLE_READ	Si se leyó un registro, y otra transacción lo modifica, guardándolo, y lo volvemos a leer, seguiremos viendo la información que había cuando leímos por primera vez.
TRANSACTION_SERIALIZABLE	Se verán todos los registros tal y como estaban antes de comenzar la transacción, no importa las modificaciones, ni lo hemos leído antes o no. Si se añadió algún nuevo registro, tampoco se verá.

```
    select.setString(1,
      DIS_ID_DISCO[i]);
    resultado = select.
      executeUpdate();
}
```

TRANSACCIONES

Por defecto una conexión funciona en modo *autocommit,* cuando se abre y se cierra automáticamente una transacción que sólo afecta a dicha sentencia. Esta opción puede ser modificada mediante el método *setAutocommit* de la interfaz *Connection.* El método *getAutoCommit* indica si estamos en modo *autocommit* o no.

Si no trabajamos en modo *autocommit* es necesario cerrar explícitamente las transacciones mediante la sentencia *commit* si tienen éxito, o *rollback,* si fallan.

```
public void commit() throws
    SQLException
```

Este método hace que todos los cambios que se han producido desde el último *commit/rollback* sean permanentes. De esta forma se liberan además todos los recursos que la base de datos mantiene sobre la conexión.

```
public void rollback() throws
    SQLException
```

Este método deshace todos los cambios que se han producido desde el último *commit/rollback.*

Cada vez que se cierra una transacción, la próxima vez que se ejecute una sentencia *SQL* se abrirá automáticamente una nueva.

Figura 4.- Resultado del *servlet* que muestra el contenido de la base de datos y permite introducir nuevos datos.

Es posible especificar el nivel de aislamiento de una transacción, mediante el método *setTransactionIsolation*, así como averiguar cuál es el nivel de aislamiento actual mediante el método *getTransaction Isolation.*

```
public abstract int
    getTransactionIsolation()
    throws SQLException
public abstract void set
    TransactionIsolation
    (int level) throws
    SQLException
```

Las características de los datos es proporcionada por ResultSetMetaData

Los niveles de aislamiento de representan mediante constantes. Así lo vemos en la Tabla 1.

LA INTERFAZ RESULTMETADATA

Esta interfaz permite conocer el tipo de cada campo o columna, su nombre, si se autoincrementa, si admite valores nulos, etc. El método *getMetaData* de la interfaz *ResultSet* permite crear un objeto de tipo *ResultSetMetaData.*

```
public abstract ResultSetMetaData
    getMetaData() throws
    SQLException
```

A continuación se recopilarán algunos de los métodos más impor-



tantes que ayudarán a conocer mejor las características de las tablas de la base de datos.

Podemos conocer el número de columnas devuelto por la consulta, así como el nombre de las mismas y el tipo *Java SQL* invocando a los métodos *getColumnCount, getColumnName, getColumnType* y *getColumnTypeName.*

```
public abstract int
     getColumnCount() throws
     SQLException
public abstract String
     getColumnName(int column)
     throws SQLException
public abstract int
     getColumnType(int column)
     throws SQLException
public abstract String
     getColumnTypeName(int column)
     throws SQLException
```

El tipo de las columnas puede obtenerse como un número entero o bien como una cadena de texto, que evidentemente se corresponde con el nombre del tipo en cuestión. Los tipos *SQL* se encuentran en la clase *java.sql.Types.*

También es posible consultar algunas características de los datos que se guardan en cada columna.

```
public abstract boolean
     isSigned(int column) throws
     SQLException
public abstract int
     isNullable(int column) throws
     SQLException
public abstract boolean
     isAutoIncrement(int column)
     throws SQLException
public abstract boolean
     isReadOnly(int column) throws
     SQLException
public abstract boolean
     isWritable(int column) throws
     SQLException
```

El método *isSigned* devuelve *True* en el caso de que la columna almacene números con signo. El método *isNullable* devuelve un número entero que indica si la columna puede contener un valor igual a *Null.* Puede ser uno de los tres siguientes valores:

```
ResultSetMetaData.columnNoNulls
     No se permiten valor nulos.
ResultSetMetaData.columnNullable
     Se permiten valores nulos.
ResultSetMetaData.column
     NullableUnknown
     Se desconoce si se permiten o
              no valores nulos.
```

El método *isAutoIncrement* devuelve *True* en el caso de que la columna sea del tipo autoincremento. Por último, dentro de este grupo se encuentran los métodos *isReadOnly* y *isWritable* que devuelven *True* o *false* dependiendo si la columna puede actualizarse.

Otros dos métodos que resultan muy útiles son *getColumnDisplaySize* y *getPrecision.*

```
public abstract int getColumn
     DisplaySize(int column)
     throws SQLException
public abstract int get
     Precision(int column) throws
     SQLException
```

El primero de ellos devuelve la anchura máxima en caracteres, necesaria para mostrar el contenido de la columna. El segundo, el número de dígitos que componen la columna.

El método commit se utiliza para marcar aquellas transacciones que sean válidas

Podemos modificar los ejemplos que hemos desarrollado hasta ahora para acceder a la interfaz *ResultetMetaData.* Así por ejemplo:

```
out.println("<TR>");
out.println("<TD ALIGN=\"right\"
     VALIGN=\"top\">getColumn
     Name</TD>");
out.println("<TD ALIGN=\"right\"
     VALIGN=\"top\">  </
     TD>");
for(int i=1; i<=rmd.getCo-
     lumnCount(); i++) {
  out.println("<TD ALIGN=\"left\"
     VALIGN=\"top\">");
  out.println("" + rmd.get
     ColumnName(i) + " (" + i +
     ") ");
  out.println("</TD>");
}
out.println("<TR>");
```

El código anterior recorre las columnas que forman parte de los datos obtenidos por una consulta a la base de datos y obtiene el nombre de las mismas.

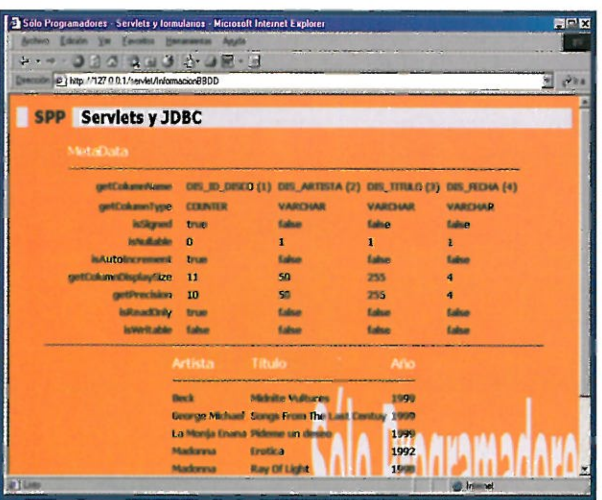
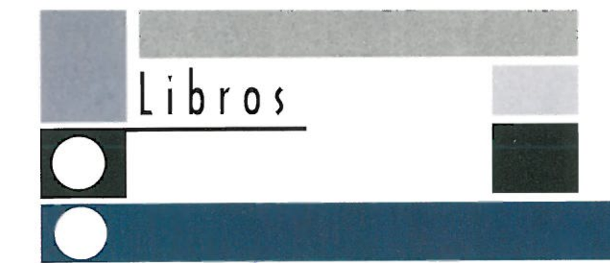


Figura 5.- Ejemplo de utilización de la interfaz *ResultSetMetaData* para obtener información acerca de la base de datos.



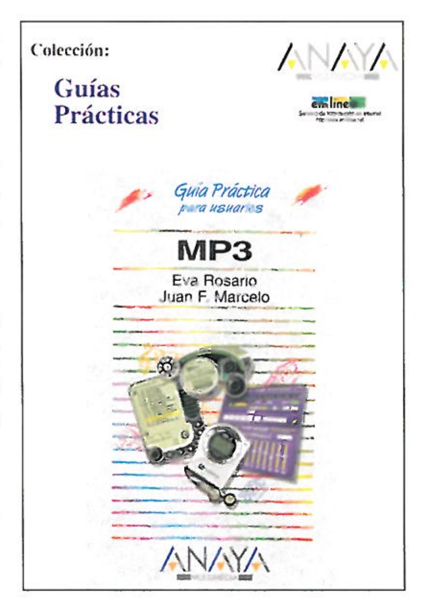
MP3

El desarrollo de los archivos *MP3* ha venido a convulsionar toda la estructura de la industria discográfica y del mercado de la música. ¿Han perdido el control las grandes compañías?

Para los que todavía no conozcan este nuevo formato, en la presente publicación, *Eva Rosario* y *J.F. Marcelo* recorren el mundo del *MP3*, desde el concepto mismo, a sus expectativas para el futuro, pasando por las tiendas en *Internet* y los programas para *DJ's*.

Esta Guía Práctica es una obra de referencia, al tiempo que una fuente de aprendizaje, y será sin duda, de utilidad tanto para los aficionados que quieran iniciarse en las peculiaridades de este formato de compresión, como para los curiosos e interesados en conocer los secretos de este nuevo sistema de comercialización a través de la red.

El libro viene acompañado de un *CD-ROM* con *demos* e interesante *shareware* y *freeware*. Imprescindible para melómanos.



Editorial: Anaya Multimedia
Número de páginas: 288
Niveles: Básico-medio
Autores: Eva Rosario y J. F. Marcelo
Idioma: Español
Precio: 1.995 Ptas. (I.V.A inc.)

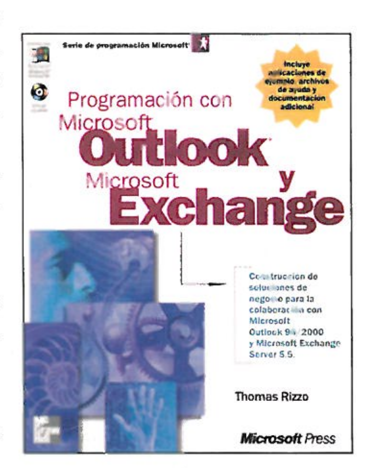
PROGRAMACIÓN CON MICROSOFT OUTLOOK Y MICROSOFT EXCHANGE

Con *Outlook* y *Exchange, Microsoft* pone a disposición de la empresa un conjunto de herramientas para ampliar y asegurar su comunicación, almacenar la información e implementar nuevas tecnologías. Son herramientas de gestión, seguimiento y servicio, dispuestas para la integración de diferentes tecnologías en un mismo entorno.

La utilización oportuna de estas herramientas de tipo cliente-servidor, permite al usuario aprovechar adecuadamente las inversiones realizadas y optimizar las futuras.

Esta obra nos guiará a través del proceso de revisión de bloques constructores clave de *Outlook*, creación de aplicaciones guiadas, extensión del alcance de las aplicaciones en *Web*, y en definitiva, de la construcción de soluciones de colaboración.

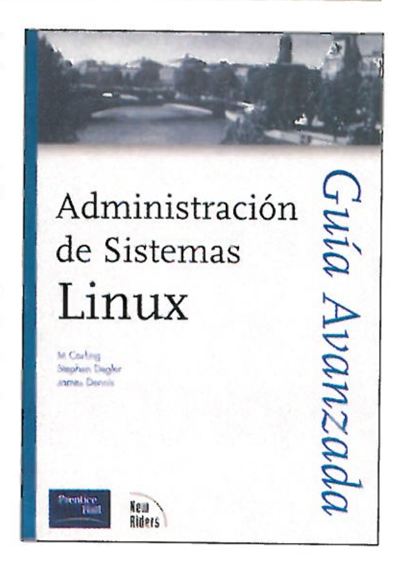
Con el libro se suministra un *CD-ROM* con ejemplos y archivos de ayuda para *Outlook, Exchange y VBScript*.



Editorial: McGraw Hill
Nº de páginas: 645
Nivel: Avanzado
Autor: Thomas Rizzo
Idioma: Español
Precio: 7.500 Ptas. (I.V.A inc.)

ADMINISTRACIÓN DE SISTEMAS LINUX

Esta Guía Avanzada, completa y clara, es una referencia para los administradores experimentados, así como un tutor para los neófitos en la ciencia y arte de la administración de uno de los sistemas en red más grandes y complejos en continua evolución.



Esta ambivalencia, unida al afán por presentar un conjunto coherente, sin perderse en el detalle, lleva a sus autores a calificar la obra como "libro no técnico dirigido a los técnicos".

El libro consta de tres partes: en la primera, se ofrece una explicación, accesible a cualquiera, de la labor de los administradores de sistemas y las peculiaridades de *Linux*; en las otras dos, se profundiza en los aspectos más prácticos y técnicos de un sistema que en los últimos meses está experimentando una progresión indiscutible.

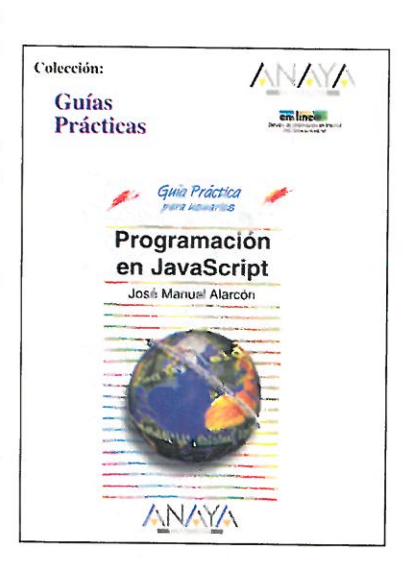
En "Administración de sistemas *Linux*", los administradores avanzados encontrarán todo lo necesario para implementar, configurar y asegurar este sistema en su empresa.

Editorial: Pearson Educación
Número de páginas: 327
Nivel: Avanzado

Autores: Carling/Degler/Dennis
Idioma: Español
Precio: 4.500 Ptas. (I.V.A. inc.)

PROGRAMACIÓN EN JAVASCRIPT

En esta ocasión, *Anaya* nos presenta una guía dedicada a todos los que quieran aprender a utilizar un lenguaje de programación imprescindible en la *Web*, aunque desconozcan cualquier otro lenguaje destinado a este fin.



JavaScript es un subconjunto de *Visual Basic* que no agota sus posibilidades en la adición de contenidos dinámicos o programáticos en páginas *Web*, si no que nos permite, además, alterar y personalizar un gran número de aplicaciones.

En esta guía se explica, paso a paso, la utilización de este lenguaje como un código que nos permita trabajar en la práctica totalidad de navegadores de *Internet*, al tiempo que se señalan sus principales diferencias dependiendo de cada versión.

Variables y funciones, operadores, programación orientada a objetos, matrices, formularios y controles *HTML*, *cookies*, sitios *Web* de interés... todo lo necesario para adentrarse en el lenguaje *JavaScript*.

La Guía Práctica de *José Manuel Alarcón* faculta a cualquier usuario no experto para programar y personalizar sus propias páginas *Web*.

Editorial: Anaya Multimedia
Nº de páginas: 288
Nivel: Básico-Medio

Autores: José Manuel Alarcón
Idioma: Español
Precio: 1.700 Ptas. (I.V.A. inc.)

Dudas técnicas

En esta sección, como cada mes, SÓLO PROGRAMADORES os brinda la oportunidad de encontrar respuesta a las dudas que podáis tener en algún tema relacionado con la programación bajo cualquier entorno de desarrollo.

PREGUNTA

Hola amigos de *Sólo Programadores*.

Estoy siguiendo la serie sobre *Servlets* y *JDBC* desde que empezó y he pensado seriamente en utilizar los *Servlets* para mi sitio *Web*. Pero tengo algunas dudas que espero que podáis responder. Intentaré ser breve:

- En mis páginas *Web* utilizo con frecuencia *cookies*. ¿Pueden ponerse y borrarse *cookies* con los *Servlets*? ¿Cómo?

- Con los *Servlets* estoy obligado a generar las páginas *HTML* con sentencias *out.prinln* y esto es un gran inconveniente. ¿Hay alguna forma de evitarlo de manera que yo pueda editar mis páginas con un editor *HTML*?

- ¿Qué servidores permiten alojar *Servlets*?.

Espero que me respondáis lo antes posible pues tengo que tomar una decisión dentro de poco. Muchas gracias.

Miguel Sarcón.

RESPUESTA

Estimado Miguel:

```
import java.awt.*;
import java.applet.*;

public class MiApplet extends Applet {

    String cadena = null;

        public void init() {
            setLayout(null);
            setSize(0,0);
        }

        public String DevolverCadena() {
            return cadena;
        }

        public void EstablecerCadena(String s) {
            cadena = s;
        }
}
```

Figura 1.- Código fuente del *Applet* que no tiene "rostro" y será utilizado en la página *HTML* con *JavaScript*.

La *API Java Servlets* proporciona la posibilidad de manejar *cookies* mediante la utilización de la clase *Cookie*. El constructor de la clase se define de la siguiente manera:

```
public Cookie(String name, String value)
```

El primero de los argumentos es el nombre de la *cookie* y el segundo de ellos es el valor.

Además de estas características, una *cookie* se define principalmente por su caducidad. Ésta puede ser establecida con el método *setMax Age*, que se define como sigue:

```
public void setMaxAge(int expiry);
```

El parámetro *expiry* representa el número de milisegundos que durará la *cookie* desde el momento en el que se ponga. Los valores negativos expresan que la *cookie* no debe ser guardada. Se utiliza el valor 0 para expresar que se desea borrar la *cookie*.

Así establecer una *cookie* es algo tan sencillo como hacer:

```
Cookie mi_cookie = new
    Cookie("numero","123456");
mi_cookie.setMaxAge(100000);
...
resp.addCookie(mi_cookie)
```

El método *addCookie* de la interfaz *HttpServletResponse* hace que la *cookie* forme parte de la respuesta dada por el *Servlet* a la petición realizada por el cliente, normalmente, el usuario desde un navegador:

```
public void addCookie(Cookie cookie);
```

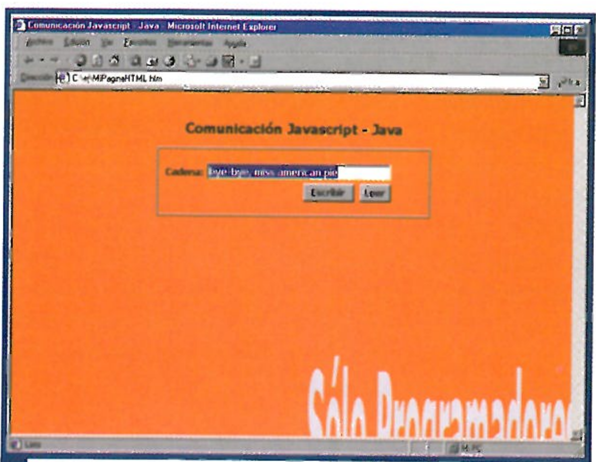


Figura 2.- Aspecto que presenta la página *HTML* en la que se produce la comunicación *JavaScript-Applet Java*.

Hay que tener en cuenta que las *cookies* utilizan las cabeceras del protocolo *HTTP*, por lo que el método *addCookie* debe llamarse en primer lugar dentro de un *Servlet* antes de realizar más cosas.

Por último, para leer *cookies* con los *Servlets* puedes utilizar el método *getCookies* de la interfaz *HttpServeltRequest*.

```
public Cookie[] getCookies();
```

Este método devuelve un *array* con todas las *cookies* disponibles para esa petición. Podemos conocer las características de cada una de ellas utilizando los siguientes métodos de la clase *Cookie*:

```
public int getMaxAge();
public String getName();
public String getValue();
```

Efectivamente, los *Servlets* no ofrecen una buena solución para generar páginas *HTML*, ya que el código de éstas tiene que ir dentro del propio *Servlet*, entremezclado con el código *Java*. Esta es una de las razones por las que *Sun* ha desarrollado lo que se conoce como *Java Server Pages* (*JSP*). Se podría decir a grandes rasgos que esta tecnología es a *Java* y a *Sun*, lo que las páginas *ASP* son a *Basic* y a *Microsoft*.

Las páginas *JSP* se basan en los *Servlets* –de hecho se podría decir que son *Servlets* "escritos de otra forma"- y permiten una gestión más adecuada de la parte *HTML* de las aplicaciones *Web*. Si tu servidor no es capaz de ejecutar páginas *JSP*, entonces quizás deberías recurrir a desarrollar un sistema de plantillas propio, de forma que los *Servlets* leyeran plantillas de *HTML* en vez de generar ellos mismos el código.

En cuanto a tu última pregunta, no queda demasiado claro si te refieres a aplicaciones o a proveedores de servicios. En cualquier caso te remitimos a la siguiente dirección:

```
www.serverpages.com/Java_Server
    _Pages/
```

En ella podrás encontrar abundante y completa información acerca de los *Servlets* y de las páginas *JSP*.

PREGUNTA

Hola, gente de *Sólo Programadores*:

Me gustaría saber si es posible hacer que dentro de una página *Web* se pueda manejar un *Applet* con *JavaScript*. ¿Cómo podría hacerlo? Quiero hacer un *Applet* que el usuario no pueda ver y que se conecte a otra *URL* para obtener unos datos. Después de procesarlos, parte de esos de datos procedentes de la conexión deben aparecer dentro de la página en varios campos de texto de un formulario. ¿Cómo podría ocultar el *Applet*?

Muchas gracias, ánimo y seguid así.

Luis Martín.

RESPUESTA

Estimado Luis:

Efectivamente, no hay ningún problema en llevar a cabo esto que nos planteas en tu pregunta. La comunicación entre un *Applet* y *JavaScript* puede ser muy interesante para afrontar aplicaciones *Web* complejas, y en la práctica resulta bastante sencillo de hacer. Veamos un pequeño ejemplo:

Hemos desarrollado un *Applet* simple que no hace nada más que escribir y leer una cadena de texto, que se almacena como miembro de la clase.

```
import java.awt.*;
import java.applet.*;

public class MiApplet extends
    Applet {
  String cadena = null;
  ...
}
```

El método *init* del *Applet* es tan sencillo como hacer:

```
public void init() {
  setLayout(null);
  setSize(0,0);
}
```

Esto es debido a que el *Applet* no tiene "rostro". Se trata solamente de un módulo de *software* que proporciona una interfaz que será manejada con *JavaScript*, pero que en ningún caso muestra nada dentro de la página *HTML*.

Los dos métodos del *Applet* que serán utilizados desde la página se definen como sigue:

Figura 3.- Código fuente de la página *HTML* en el que se produce la comunicación *JavaScript-Applet Java.*

```
public String DevolverCadena() {
  return cadena;
}
public void
    EstablecerCadena(String s) {
  cadena = s;
}
```

El método *DevolverCadena* devuelve la cadena de texto almacenada como miembro de la clase, y el método *EstablecerCadena* le asigna un nuevo valor.

Este *Applet* se incluye dentro de la página *Web* utilizando la etiqueta *<APPLET>*.

```
<APPLET CODE="MiApplet.class"
        NAME="MiApplet"
        WIDTH="0"
        HEIGHT="0">
</APPLET>
```

Como los atributos *HEIGHT* y *WIDTH* son 0, el usuario no verá nada dentro de la página *Web*.

Para probar este *Applet* se ha creado un pequeño formulario *HTML* con un campo de texto y dos botones, etiquetados como **Escribir** y **Leer**. Cuando se pulsa el primero de ellos se toma la cadena de texto que se encuentra dentro del campo y se pasa al *Applet*. El segundo de los botones realiza la operación contraria. Es decir, lee la cadena de texto procedente del *Applet* y la escribe en el campo de texto del formulario *HTML*.

```
<INPUT TYPE="button"
       NAME="ESCRIBIR"
       VALUE="Escribir"
       onclick=
       "Escribir
       Cadena(this.form
       )">
<INPUT TYPE="button"
       NAME="LEER"
       VALUE="Leer"
onclick="LeerCadena(this.form)">
```

Las funciones *JavaScript EscribirCadena* y *LeerCadena* son las que llaman a las funciones correspondientes del *Applet,* y su código es el siguiente:

```
function LeerCadena(f) {
  f.CADENA.value = document.
    MiApplet.DevolverCadena();
  f.CADENA.focus();
  f.CADENA.select();
}

function EscribirCadena(f) {
  document.MiApplet.Establecer
    Cadena(f.CADENA.value);
}
```

Como se puede observar la forma de acceder al *Applet* consiste en utilizar el valor del atributo **NAME** de la etiqueta *APPLET*> como identificador del objeto. Éste depende a su vez del objeto *document*.

PREGUNTA

Hola, *Sólo Programadores*:

Estoy desarrollando un programa con *Visual Basic* y en un momento determinado tengo que averiguar si en un directorio existe o no un fichero, ya que dependiendo de esto hago una cosa u otra. ¿Me podríais decir cómo puedo hacer esto?

Aparte de lo anterior me gustaría felicitaros por vuestra estupenda publicación, y de paso me gustaría que me dieseis buenas direcciones de *Internet* relacionadas con *Visual Basic*.

Cristina Colón.

RESPUESTA

Estimada Cristina:

Afortunadamente tu problema tiene una solución bastante rápida. En *Visual Basic* comprobar la existencia de un fichero es algo tan sencillo como verificar si se produce algún error al abrirlo. Para ello podemos utilizar la sentencia *On Error Resume Next*. La función que sigue podría servirte para tu propósito:

```
Public Function Existe(fich As
    String) As Boolean
  On Error Resume Next
  Open fich For Input As #1
  If Err Then
    Existe = False
    Exit Function
  End If
  Close #1
  Existe = True
End Sub
```

En cuanto a las direcciones *Web* que nos pides, aquí te escribimos algunas que no están nada mal:

- Visual Basic Zone
  www.vb-zone.com
- VB Helper
  www.vb-helper.com
- El Guille
  guille.costasol.net

Universe Linux™

La alternativa que su empresa estaba esperando

Linux está en boca de todos: potencia, versatilidad, seguridad con mayúsculas, economía. Y todos se hacen las mismas preguntas. ¿Debo cambiar a **Linux**? ¿Es cierto que **Linux** es más fiable en la empresa que Windows NT, 98 o Windows 2000? ¿Es cierto que **Linux** puede convivir con Windows intercambiando información de forma transparente? ¿Puedo ahorrar con soluciones basadas en **Linux** hasta el 60% del precio en sus programas equivalentes de Microsoft? ¿Por qué más del 65% de los servidores de Internet corren bajo **Linux**? ¿Quién me dará el soporte y la garantía necesarios?
La respuesta a todas las preguntas se llama ***Universe Linux***

Universe Linux es la primera empresa de nuestro país dedicada exclusivamente a ofrecer soluciones basadas en **Linux** para usuarios y empresas: distribuciones **Linux**, utilidades, consultoría, instalaciones a medida, mantenimiento, formación... Todo lo que Vd. necesite y al mejor precio del mercado.

Puede que aún no sepa lo que ***Universe Linux*** puede hacer por Vd. Le invitamos a conocer nuestros productos con una demostración gratuita en nuestra sucursal de Madrid. Llámenos para concertar la demostración.
Infórmese sin compromiso llamando al **91-356 69 08** o visitando nuestra página web:

http://www.u-linux.com

Llame hoy mismo. *Por fin* en LINUX hay alguien que RESPONDE.

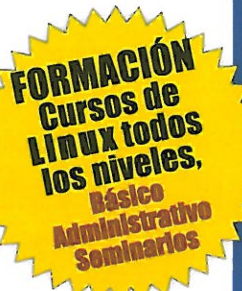
Linux PYMES

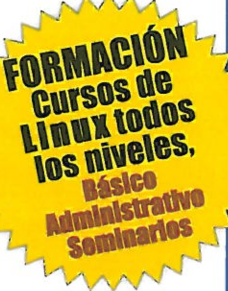
La distribución pensada para empresas
Pvp. recomendado 9.900 pts.

Por fin, la primera distribución creada para las PYMES: robusta, fiable, sencilla de instalar. Incluye: StarOffice, escritorio KDE, manual de instalación paso a paso y, como oferta de lanzamiento, GRATIS el programa LINUX FAX SERVER, valorado en 14.900 pts. (*)

Características:
- Instalación en modo gráfico con pantallas de ayuda
- Configuración X completa antes de la instalación de paquetes
- Flexibilidad en los modos de instalación Server y Workstation
- Utilidades de automontaje
- XFree 3.3.5-3, Kernel 2.2.12-20, KDE 1.1.2, GTK, etc.

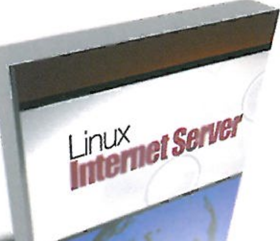
(*) Oferta válida hasta el 1 de abril del 2000.






Linux INTERNET SERVER

El servidor más fiable
Pvp. recomendado 49.950 pts.



El servidor de Internet pensado para su empresa incluye:
***Servidor de correo electrónico, páginas Web y FTP.**
***Servidor de proxy y caché.**
***Gestión y mantenimiento de listas de correo.**
***Alojamiento del Dominio y Dirección IP fija.**
***Gestión de usuarios y autentificación de los mismos en la entrada al sistema.**
***Control y visualización de usuarios conectados.**
***10 niveles de seguridad en el servidor de páginas Web local.**
***Programación de eventos de conexión con Internet.**
Ver demostración en http:// www.u-linux.com/demo

Linux FAX SERVER

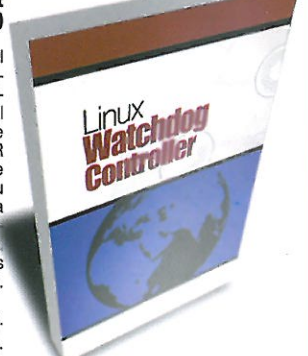
Ahorre enviando todos sus fax por la intranet
Pvp. recomendado 24.900 pts.

Se acabaron las colas para enviar un fax. El primer servidor de fax para Linux selecciona entre los tramos horarios más económicos de su operador de comunicaciones y se adapta a ellos, economizando en cada envío. Evita desplazamientos dentro de la propia empresa ahorrando tiempo y molestias al utilizar la intranet para acceder al servidor. Crea las estadísticas necesarias para analizar los gastos por cada usuario. Economía y comodidad garantizadas en sus comunicaciones.

LINUX WATCHDOG CONTROLLER

El guardián de su empresa en Internet
Pvp. recomendado 69.950 pts. (*)

El primer sistema de CONTROL TOTAL sobre Internet en su empresa. Limitación en el tiempo de conexión, listas de direcciones, chat, en general los puertos más representativos del sistema. Estadísticas de conexión por usuario, por sites accedidas, por distribución del tiempo según operador. Limitaciones individuales o por grupos de usuarios. El sistema corta la conexión cuando detecta que el usuario ha sobrepasado los límites de sus privilegios y le envía un mensaje al supervisor. LINUX WATCHDOG CONTROLLER es una herramienta imprescindible en la empresa moderna para el control del uso de Internet. El 90% de los empleados de las empresas con acceso a Internet desde su puesto de trabajo navegan con fines personales: chats, compras, etc. Esta herramienta erradica el problema realizando auditorías individualizadas de cada usuario, ya que almacena direcciones accedidas, tiempo de estancia, información recibida y enviada, etc. LINUX WATCHDOG CONTROLLER es una utilidad imprescindible para los Administradores del Sistema y Jefes de Personal. Pruébelo sin compromiso.



(*) Hasta 10 puestos de trabajo. Consultar según necesidades superiores.
Los precios no incluyen IVA.



Universe Linux *le brinda las mejores opciones para* **hacer negocio:**

Linux Training Center

Formación a todos los niveles

Si dispone de una academia de informática Vd. puede tener el LINUX TRAINING CENTER oficial de su población.
Universe Linux formará a su personal y le proporcionará todo el material didáctico necesario y la certificación LINUX TRAINING CENTER. Además, le incluiremos en nuestra Lista de Centros Autorizados en todas las acciones publicitarias promocionando su negocio. (*)
La más amplia gama de cursos para todos los niveles y necesidades en formación: Introducción a Linux, Administración de Sistemas, Instalación y configuración de Servidores web, Ofimática en Linux, etc.
Sistema de franquicias con número limitado de licencias por provincia.

(*) Más de 250.000 impactos cada mes en prensa especializada y general.

Linux Solution Provider

Una apuesta segura para emprendedores. Servicios en Linux

Si es Vd. socio o propietario de una empresa de Servicios Informáticos puede obtener para su provincia la representación en *exclusiva* de los productos **Universe Linux** y la certificación **LINUX SOLUTION PROVIDER** para su empresa de servicios, distribuyendo, instalando y dando soporte a la más amplia gama de productos Linux, con los mejores márgenes comerciales del mercado. Además, contará con el asesoramiento de nuestros técnicos y ayuda on line.
Un negocio de rentabilidad asegurada, sin cánones de entrada, compatible con su actividad y clientes habituales. No deje pasar la oportunidad e infórmese en el **91-356 69 08.**

Linux Developer

La mejor opción para desarrolladores

Universe Linux ofrece a los profesionales independientes del mundo de la programación la posibilidad de adquirir todos los conocimientos para trabajar en el mundo Linux. Incluye el material didáctico y el software necesario a un precio excepcional. Sólo para desarrolladores.

Y también: STAROFFICE, APPLIXWARE, ANTIVIRUS, bases de datos relacionales, servidores de ficheros, y mucho más.

Distribuidores

Universe Linux S.L. España ofrece las mejores condiciones para distribuidores, con descuentos desde el 25% hasta el 40% según tramos de compra. Infórmese sin compromiso en el teléfono:
(91) 356 69 08.
Si desea solicitar una demostración gratuita de alguno de los programas, póngase en contacto con nosotros.

Linux S.L • C/ Conde de Vilches 21 • 28028 MADRID • Teléfono: (91) 356 69 08 • Fax: (91) 355 28 95 • ***http://www.u-linux.com***



Su empresa
Tiene

Aplicaciones desarrolladas bajo Microsoft Windows

"Nuestra empresa ha desarrollado una aplicación. ¿Cuál será el sitio donde la puedan conocer el mayor número de clientes potenciales?"

La solución la encontrará en

msdn.microsoft.es/catalogo

CATÁLOGO DE SOLUCIONES Microsoft **Windows**

El lugar donde se encuentra el mayor número de aplicaciones desarrolladas bajo Microsoft Windows para cualquier tipo de empresas.

Su empresa
necesita

Aplicaciones de desarrollo bajo Microsoft Windows

"Buscamos una aplicación de gestión a medida para la actividad de nuestra empresa"

Las 50 primeras soluciones* que se incluyan en nuestro catálogo de soluciones, obtendrán un ratón IntelliMouse Explorer o un teclado Microsoft Internet Keyboard

* Solamente se obtendrá un regalo por empresa

Microsoft